Internet Engineering Task Force (IETF) L. Lhotka, Ed.
Request for Comments: 6110 CESNET
Category: Standards Track February 2011
ISSN: 2070-1721

# Mapping YANG to Document Schema Definition Languages and Validating NETCONF Content

Сопоставление YANG с языками DSDL и проверка содержимого NETCONF

**Аннотация**

Этот документ задаёт правила отображения моделей данных YANG в языки определения схемы документа (Document Schema Definition Languages или DSDL) - скоординированный набор языков схем XML, стандартизованный как ISO/IEC 19757. Рассмотрены сопоставления с языками Regular Language for XML Next Generation (RELAX NG), Schematron и Document Schema Renaming Language (DSRL). Отображение принимает один или несколько модулей YANG и создаёт набор схем DSDL для выбранного типа целевого документа - хранилища содержимого сообщения протокола настройки сети (Network Configuration Protocol или NETCONF) и т. п. Рассмотрены также процедуры проверки таких документов на основе схем.

**Статус документа**

Документ относится к категории Internet Standards Track.

Документ является результатом работы IETF[1] и представляет согласованный взгляд сообщества IETF. Документ прошёл открытое обсуждение и был одобрен для публикации IESG[2]. Дополнительную информацию о стандартах Internet можно найти в разделе 2 в RFC 5741.

Информация о текущем статусе документа, найденных ошибках и способах обратной связи доступна по ссылке



# Оглавление



[1]Internet Engineering Task Force - комиссия по решению инженерных задач Internet.
[2]Internet Engineering Steering Group - комиссия по инженерным разработкам Internet.

# *1. Введение*

Рабочая группа NETCONF завершила спецификацию базового протокола управления конфигурацией [RFC4741]. Эта спецификаци задаёт протокольные привязки и синтаксис контейнеров XML для операций настройки и управления, но не включает язык моделирования данных или соответствующие правила для передачи модели конфигурации и данных состояния протоколом NETCONF. Направление IETF Operations Area имеет давние традиции определения данных для модулей баз управляющей информации (Management Information Bases или MIB) [RFC1157] простого протокола управления сетью Simple Network Management Protocol или SNMP) с использованием языка структур управляющей информации (Structure of Management Information или SMI) [RFC2578] для моделирования данных. Хотя такой подход к моделирования имеет множество известных проблем, большая часть функций моделирования данных в SMI по-прежнему считается очень важной. Простое моделирование допустимого синтаксиса бех дополнительных семантических связей в прошлом вызывало многочисленные проблемы совместимости.

Сообщество NETCONF пришло к выводу о необходимости схемы моделирования данных для поддержки текущих разработок IETF и задаваемых производителями модулей управляющей информации. Была создана рабочая группа NETMOD для разработки языка моделирования, определяющего семантику рабочих данных, уведомлений о событиях и операций с упором на удобство для человека, т. е. удобочитаемость и простоту использования. Результатом стал язык моделирования данных YANG [RFC6020], который сейчас служит для нормативного описания моделей данных NETCONF.

Поскольку NETCONF использует XML для кодирования своих сообщений, естественно выражать ограничения на содержимое NETCONF с помощью стандартных языков схем XML. Для этого рабочая группа NETMOD WG выбрала DSDL, стандартизированные как ISO/IEC 19757 [DSDL]. Платформа DSDL включает набор языков схем XML, учитывающих правила грамматики, семантические ограничения и другие аспекты моделирования данных, делая это согласованным и скоординированным способом, что очень важно. Хотя некоторые части DSDL ещё не стандартизованы и работа продолжается, три части, на которые опирается сопоставление YANG-DSDL - Regular Language for XML Next Generation (RELAX NG), Schematron и Document Schema Renaming Language (DSRL) - уже имеют статус ISO/ IEC International Standard и поддерживаются множеством программных инструментов.

Этот документ содержит спецификацию отображения, которое транслирует модели данных YANG в схемы XML с применением подмножества языков схем DSDL. Процедура сопоставления делится на два шага. Сначала структура дерева данных, подписи удалённых вызовов процедур (remote procedure call или RPC) и уведомления выражаются в так называемой гибридной схеме - одной схеме RELAX NG с аннотациями, представляющими дополнительные сведения модели данных (метаданные, документацию, семантические ограничения, принятые по умолчанию значения и т. п.). На втором шаге создаётся скоординированный набор схем DSDL, которые можно использовать для проверки конкретных документов XML, таких как запросы клиентов или уведомления, возможно, с учётом дополнительного контекста, такого как активные возможности или функции.

# *2. Термины и обозначения*

Ключевые слова **должно** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **не рекомендуется** (NOT RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с [RFC2119].

Ниже перечислены термины, заданные в [RFC4741]:

- client - клиент;
- datastore - хранилище данных;
- message - сообщение;
- operation - операция;
- server - сервер.

Ниже перечислены термины, заданные в [RFC6020]:

- augment - дополнение;
- base type - базовый тип;
- built-in type - встроенный тип;
- configuration data - данные конфигурации;
- container - контейнер;
- data model - модель данных;
- data node - узел данных;
- data tree - дерево данных;
- derived type - производный тип;
- device deviation - отклонение устройства;
- extension - расширение;
- feature - свойство, функция;
- grouping - группировка;
- instance identifier - идентификатор экземпляра;
- leaf-list - лист-список;
- list - список;
- mandatory node - обязательный узел;
- module - модуль;
- RPC - удаленный вызов процедуры;
- RPC operation - операция RPC;
- schema node - узел схемы;
- schema tree - дерево схемы;
- state data - данные состояния;
- submodule - субмодуль;
- top-level data node - узел данных верхнего уровня;
- uses - использует.

Ниже перечислены термины, заданные в [XML-INFOSET]:

- attribute - атрибут;
- document - документ;
- document element - элемент документа;
- document type declaration (DTD) - объявление типа документа;
- element - элемент;
- information set - набор информации;
- namespace - пространство имён.

В тексте применяются перечисленные ниже типографические соглашения:

- ключевые слова операторов YANG указываются в одинарных кавычках[1];
- имена элементов XML указываются в угловых скобках <>;
- имена атрибутов XML указываются с префиксом @;
- иные буквальные значения указываются в двойных кавычках[1].

Имена элементов XML всегла указываются с явным префиксом пространства имён, соответствующим указанным ниже словарям:

a - аннотации совместимости DTD [RNG-DTD];

dc - элементы метаданных Dublin Core [RFC5013];

dsrl - язык переименования семантики документов (Document Semantics Renaming Language) [DSRL];

en - уведомления NETCONF о событиях [RFC5277];

nc - протокол NETCONF [RFC4741];

nma - относящиеся к NETMOD аннотации схем (5.3. Связанные с NETMOD аннотации);

nmf - относящиеся к NETMOD функции расширения языка путей XML (XML Path Language или XPath) (12.7. Аннотация <nma:instance-identifier>);

rng - RELAX NG [RNG];

sch - ISO Schematron [Schematron];

xsd - W3C XML Schema [XSD].

В таблице 1 указаны сопоставления этих префиксов с URI пространств имён.

*Таблица 1. Префиксы и URI пространств имен.*

| ***Префикс*** | ***URI пространства имен*** |
|---|---|
| a | http://relaxng.org/ns/compatibility/annotations/1.0 |
| dc | http://purl.org/dc/terms |

[1]В переводе это не используется. *Прим. перев.*

| | |
|---|---|
| dsrl | http://purl.oclc.org/dsdl/dsrl |
| en | urn:ietf:params:xml:ns:netconf:notification:1.0 |
| nc | urn:ietf:params:xml:ns:netconf:base:1.0 |
| nma | urn:ietf:params:xml:ns:netmod:dsdl-annotations:1 |
| nmf | urn:ietf:params:xml:ns:netmod:xpath-extensions:1 |
| rng | http://relaxng.org/ns/structure/1.0 |
| sch | http://purl.oclc.org/dsdl/schematron |
| xsd | http://www.w3.org/2001/XMLSchema |

## 2.1. Новые термины

***ancestor data type - тип данных предка***

Любой тип данных из которого (транзитивно) выведен этот тип.

***ancestor built-in data type - встроенный тип данных предка***

Встроенный тип, с которого начинается цепочка производных типов для этого типа данных.

***hybrid schema - гибридная схема***

Схема RELAX NG с аннотациями, которая содержит те же сведения, что и исходный модуль или модулия YANG (8.1. Гибридная схема).

***implicit node - неявный узел***

Узел данных, который не будучи созданным в дереве данных, может быть добавлен в набор информации этого дерева (конфигурация, ввод или вывод RPC, уведомление) без изменения семантики дерева данных.

# *3. Цели и мотивация*

Основной целью этой работы является дополнение языка моделирования данных YANG возможностями проверки языков схем DSDL - RELAX NG, Schematron, DSRL. Документ описывает соответствия между грамматикой, семантикой и ограничениями типов данных в YANG и эквивалентными шаблонами и правилами DSDL. Конечной целью является возможность сбора всех значимых сведений из модулей YANG и их представления в схемах DSDL. Хотя отображение YANG в DSDL, описанное в документе, в принципе может быть обратимым, отображение DSDL в YANG выходит за рамки этого документа.

Основанные на XML модели данных и кодирование данных в XML присутствуют в нескольких формах на разных этапах моделирования YANG и рабочего процесса NETCONF - содержимое хранилищ данных конфигурации, запросы и отклики RPC, уведомления. Кроме того, операции RPC отличаются разнообразием, обусловленным выборочной доступностью возможностей и функций. Модули YANG могут определять новые операции RPC. Отображениям следует поддерживать такую изменчивость и создавать схемы, адаптированные к конкретной ситуации и поэтому более эффективные для проверки, нежели базовые всеобъемлющие схемы. Для охвата такой изменчивости предлагается генерировать схемы DSDL по запросам для конкретной задачи из доступного набора модулей YANG. Жихненный цикл таких схем будет сравнительно коротким, иными словами, не предполагается, что какой-либо набор схем DSDL будет создаваться для долгосрочной поддержки вместе с модулями YANG.

Сгенерированные схемы предназначены в основном для использования в качестве входных данных для имеющихся средств проверки схем XML и других готовых инструментов. Однако схемы могут внимательно рассматривать разработчики и пользователи как формальное представление ограничений для конкретных объектов данных в коде XML. Поэтому вторая цель состоит в сохранении удобочитаемости схем и в результате сопоставления разбивается на две части.

1. На первом этапе 1 или несколько модулей YANG отображается в так называемую гибридную схему, которая представляет собой одну схему RELAX NG, описывающую грамматические ограничения для основного дерева данных, а также операций RPC и уведомлений. Семантические ограничения и другие сведения из входных модулей YANG записываются в гибридную схему в форме аннотаций внешних пространств имён. Вывод этого этапа можно считать практически полным эквивалентом входных модулей YANG. Однако он не подходит напрямую для какой-либо проверки.

2. На втором этапе полученная гибридная схема преобразуется в согласованный набор схем DSDL, содержащих ограничения для конкретных объектов данных и ситуаций. Эти схемы DSDL предназначены, прежде всего, для машинной проверки с использованием готовых инструментов.

# *4. Языки схем DSDL*

DSDL - это набор языков схем, разрабатываемых как международный стандарт ISO/IEC 19757 [DSDL]. В отличие от других подходов к проверке документов XML, прежде всего W3C XML Schema Definition (XSD) [XSD], модель DSDL придерживается принципа «малых языков» - каждый компонент DSDL представляет собой законченный язык схем с относительно небольшой сферой применения (назначением). Совместно эти языки можно применять для согласованного решения различных задач проверки.

Описанное в этом документе отображение использует три языка схем DSDL - RELAX NG [RNG], Schematron [Schematron], DSRL [DSRL].

## 4.1. RELAX NG

RELAX NG (произносится как relaxing) - это язык схем XML для основанной на грамматике проверки и часть 2 в семействе стандартов ISO/IEC DSDL [RNG]. Подобно XSD, он может описывать ограничения для структуры и содержимого документов XML. Однако, в отличие от языков схем DTD [XML] и XSD, язык RELAX NG намеренно избегает каких-либо дополнений, вроде задания принятых по умолчанию значений. default values. В архитектуре DSDL конкретная задача определения и применения принятых по умолчанию значения отдана доугому языку описания схем - DSRL (4.3. DSRL).

В качестве базовой библиотеки типов данных RELAX NG использует W3C XML Schema Datatypes [XSD-D], но в отличие от XSD, можно применять и другие библиотеки типов совместно с этой библиотеков или вместо неё.

Язык RELAX NG свободно воспринимает аннотации из других пространств имён. С некоторыми исключениями такие аннотации могут помещаться в любое место схемы и не требуют инкапсулирующих элементов, таких как <xsd:annotation> в XSD.

Схемы RELAX NG можно представлять в двух вариантах - компактном и XML. Компактный синтаксис описан в Приложении C к спецификации RELAX NG [RNG-CS], добавленной в стандарт в 2006 г. (Amendment 1). Автоматические двухсторонние преобразования между двумя вариантами синтаксиса возможны с использованием нескольких инструментов, например, Trang [Trang].

Лаконичность и удобочитаемость компактного синтаксиса часто делают его предпочтительным для публикации схем RELAX NG, тогда как средства проверки и другие программные инструменты обычно работают с синтаксисом XML. Однако компактный синтаксис имеет два недостатка.

- Внешние аннотации снижают удобочитаемость схем с компактным синтаксисом. В синтаксисе XML аннотирующие элементы и атрибуты представлены простым и унифицированным способом (элементы и атрибуты XML из внешних пространств имён), а компактный синтаксис использует 4 синтаксических конструкции - документацию, грамматику, исходные и последующие аннотации. Поэтому влияние аннотаций на удобочитаемость для компактного синтаксиса зачастую гораздо сильнее нежели для синтаксиса XML.
- В компьютерных программах генерация компактного синтаксиса сложнее, чем синтаксиса XML. Имеется много библиотек для создания дерева XML в памяти и его сериализации, однако их нет для компактного синтаксиса.

По этим причинам спецификация отображения в этом документе использует лишь синтаксис XML. Однако полученные при трансляции схемы **можно** представить в эквивалентном компактном синтаксисе, где это уместно.

Элементы RELAX NG связаны с URI пространства имён http://relaxng.org/ns/structure/1.0. Пространство имён библиотеки типов данных XSD доступно по ссылке http://www.w3.org/2001/XMLSchema-datatypes.

## 4.2. Schematron

Schematron - это часть 3 DSDL, стандартизованная ISO/IEC в 2006 г. [Schematron]. В отличие от традиционных языков схем, таких как DTD, XSD, RELAX NG, которые основаны на формальной грамматике, Schematron использует основанный на правилах подход. Правила могут задавать произвольные условия, включающие данные из разных частей документа XML. Каждое правило состоит из трёх основных компонентов:

- контекст - выражение XPath, определяющее набор мест, где правило должно применяться;
- условие утверждения или отчёта - другое выражение XPath, оцениваемое в месте применения правила;
- понятное человеку сообщение, которое выводится при несоблюдении (false) условия утверждения или соблюдении (true) условия отчёта.

Разница между условиями утверждения и отчёта состоит в том, что первое выполняется, когда документ соответствует, а второе - когда возникает ошибка.

Большая часть выразительных средств Schematron взята и XPath [XPath] и XSLT[1] [XSLT]. ISO допускает привязку языка динамических запросов для возможности применения языков запросов запросов XML - STX, XSLT 1.0, XSLT 1.1, EXSLT, XSLT 2.0, XPath 1.0, XPath 2.0, XQuery 1.0 (список может быть расширен в будущем).

Понятные человек сообщения об ошибках - ещё одно отличие Schematron от других языков схем. Сообщения могут даже включать выражения XPath, оцениваемые в фактическом контексте и таким образом указывающие элементы информации в проверяемом документе XML. Друго особенностью Schematron является применение при отображениях абстрактных шаблонов, которые, по сути, работают как макросы и могут иметь параметры, подставляемые в абстрактный шаблон. Элементы Schematron связаны с URI пространства имён http://purl.oclc.org/dsdl/schematron.

## 4.3. DSRL

Язык переименования семантики документа (Document Semantics Renaming Language или DSRL, произносится disrule) - это часть 8 DSDL, стандартизованная ISO/IEC в 2008 г. [DSRL]. В отличие от RELAX NG и Schematron, языку DSRL разрешено изменять набор сведений XML проверяемого документа. Хотя DSRL предназначен в основном для переименования элементов и атрибутов XML, он может определять принятые по умолчанию значения атрибутов XML и содержимого элементов XML или ветвей дерева со вставкой принятого по умолчанию содержимого, если его нет в проверяемом документе. Последнее свойство применяется в отображении YANG-DSDL для представления принятого по умолчанию содержимого YANG в виде листьев с принятыми по умолчанию значениями и из предков - контейнеров отсутствия (non-presence). Элементы DSRL связаны с URI пространства имён http://purl.oclc.org/dsdl/dsrl.

# *5. Дополнительные аннотации*

Помимо языков схем DSDL сопоставление использует 3 набора аннотаций, добавляемых в схемы RELAX NG как атрибуты внешнего пространства имён и элементы. Два из этих наборов аннотаций - элементы Dublin Core и аннотации совместимости DTD - являются стандартными словарями для представления метаданных и документации, соответственно. Хотя эти элементы модели данных не подвергаются формальной проверке, они достаточно часто немут важные сведения для разработчиков модели данных. Поэтому их **следует** включать в гибридную схему и **можно** включать в схемы окончательной проверки. Третий набор образуют аннотации, относящиеся к NETMOD. Они предназначены для гибридных схем и содержат семантические ограничения и другие сведения, которые не могут быть напрямую выражены в RELAX NG. На втором этапе сопоставления ити аннотации преобразуются в правила Schematron и DSRL.

## 5.1. Элементы метаданных Dublin Core

Dublin Core - это система элементов метаданных, исходно предназначенная для описания метаданных ресурсов WWW с целью упрощения их поиска, а позднее принятая как стандарт описания метаданных произвольных ресурсов. В этой

[1] Extensible Stylesheet Language Transformations - трансформации языка расширяемых шаблонов стилей.

спецификации применяется определение из [RFC5013]. Элементы Dublin Core связаны с URI пространства имён http://purl.org/dc/terms.

## 5.2. Аннотации RELAX NG для совместимости DTD

Аннотации совместимости DTD являются частью спецификации the RELAX NG DTD Compatibility [RNG-DTD]. Отображение YANG-DSDL использует лишь аннотацию <a:documentation> для представления текста операторов YANG description и reference. Отметим, что нет намерения сделать полученные схемы DTD-совместимыми и основной причиной применения этих аннотаций является хорошая их поддержки и адекватное форматирование некоторыми инструментами RELAX NG. Аннотации совместимости DTD связаны с URI пространства имён http://relaxng.org/ns/compatibility/annotations/1.0.

## 5.3. Связанные с NETMOD аннотации

Специфические аннотации NETMOD являются элементами и атрибутами XML, связанными с URI пространства имён urn:ietf:params:xml:ns:netmod:dsdl-annotations:1 и присутствующие в разных местах гибридной схеиы. Операторы YANG сопоставляются с этими аннотациями напрямую. В большинстве случаев атрибуты и элементы аннотации имеют то же имя, что и соответствующий оператор YANG. В таблице 2 приведён список связанных с NETMOD атрибутов аннотаций (префикс @) и элементов (<>) со ссылками на параграфы с их описаниями. В Приложении A приведена схема RELAX NG для этого словаря аннотаций.

*Таблица 2. Аннотации, связанные с NETMOD.*

| ***Аннотация*** | ***Параграф*** |
|---|---|
| @nma:config | 10.9. Оператор config |
| <nma:data>[1] | 8.1. Гибридная схема |
| @nma:default | 10.12. Оператор default |
| <nma:error-app-tag>[2] | 10.16. Оператор error-app-tag |
| <nma:error-message>[2] | 10.17. Оператор error-message |
| @nma:if-feature | 10.22. Оператор if-feature |
| @nma:implicit | 10.11. Оператор container, 10.7. Оператор case, 10.12. Оператор default |
| <nma:input>[1] | 8.1. Гибридная схема |
| <nma:instance-identifier>[3] | 10.53.7. Тип instance-identifier |
| @nma:key | 10.26. Оператор key |
| @nma:leaf-list | 10.28. Оператор leaf-list |
| @nma:leafref | 10.53.8. Тип leafref |
| @nma:mandatory | 10.8. Оператор choice |
| @nma:max-elements | 10.28. Оператор leaf-list |
| @nma:min-elements | 10.28. Оператор leaf-list |
| @nma:module | 10.34. Оператор module |
| <nma:must>[4] | 10.35. Оператор must |
| <nma:notification>[1] | 8.1. Гибридная схема |
| <nma:notifications>[1] | 8.1. Гибридная схема |
| @nma:ordered-by | 10.38. Оператор ordered-by |
| <nma:output>[1] | 8.1. Гибридная схема |
| <nma:rpc>[1] | 8.1. Гибридная схема |
| <nma:rpcs>[1] | 8.1. Гибридная схема |
| @nma:status | 10.51. Оператор status |
| <nma:unique>[5] | 10.55. Оператор unique |
| @nma:units | 10.56. Оператор units |
| @nma:when | 10.59. Оператор when |

# *6. Обзор сопоставления*

В этом разделе дан обзор отображения YANG-DSDL (вход и выход). Общая структура приведена на рисунке 1

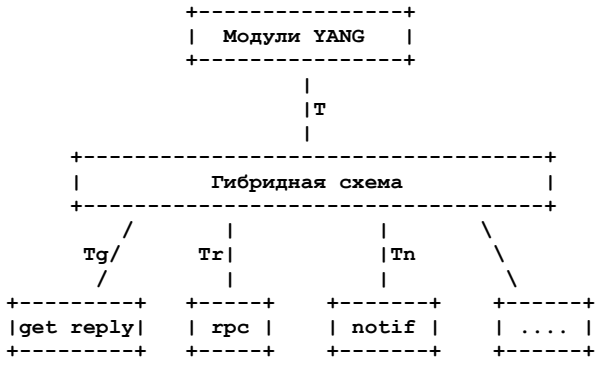


*Рисунок 1. Структура отображения.*

Процедура отображения делится на два этапа.

1. Преобразование T отображает 1 или несколько модулей YANG в гибридную схему (8.1. Гибридная схема). Ограничения, которые не выражаются напрямую в RELAX NG (определения ключей списков, must и т. п.) и документирующие тексты записываются в схему как аннотации внешних пространств имён.

[1]Маркерный элемент в гибридной схеме.
[2]Присутствует лишь как субэлемент в <nma:must>.
[3]Имеет необязательный атрибут @require-instance.
[4]Имеет обязательный атрибут @assert и два необязательных субэлемента <nma:error-app-tag> и <nma:error-message>.
[5]В оригинале ошибочно указано @nma:unique, см. https://www.rfc-editor.org/errata/eid3362. *Прим. перев.*

2. На втором этапе гибридная схема может быть разными способами преобразована в набор схем DSDL, которые могут служить для проверки отдельных объектов данных в конкретном контексте. На рисунке 1 показаны в качестве примеров 3 простых возможности. В процессе преобразования извлекаются нужные части гибридной схемы, а конкретные аннотации преобразуются в эквивалентные, но обычно более сложные шаблоны Schematron, карты элементов DSRL и т. п.

Реализация алгоритма отображения **должна** воспринимать на входе 1 или несколько действительных модулей YANG. Важна возможность совместной обработки нескольких модулей YANG, поскольку для сессии NETCONF может согласовываться несколько модулей, а содержимое хранилища данных конфигурации приходит в форме объединения деревьев данных из отдельных модулей, что может быть связано с несколькими корневыми узлами в иерархии хранилища данных. Кроме того, входные модули могут быть связаны через дополнения (augment) и в этом случае один модуль будет дополнять дерево данных другого модуля. Предполагается, что алгоритм имеет доступ (возможно, по запросу) ко всем модулям YANG, импортируемым набором входных модулей (напрямую или опосредованно).

Другие сведения из входных модулей YANG (такие как семантика ограничений и принятые по умолчанию значения) записываются в гибридную схему как аннотации - атрибуты или элементы XML, связанные с URI пространства имён urn:ietf:params:xml:ns:netmod:dsdl-annotations:1. Метаданные, описывающие модули YANG, отображаются в элементы аннотаций Dublin Core (5.1. Элементы метаданных Dublin Core), а строки документации - в элементы <a:documentation>, относящиеся к словарю соответствия DTD (5.2. Аннотации RELAX NG для совместимости DTD).

Выводом второго этапа является согласованный набор из трёх схем DSDL, соответствующих конкретному объекту данных и контексту:

- схема RELAX NG описывает ограничения для грамматики и типов данных;
- схема Schematron выражает другие ограничения, такие как неуникальность ключей списка, или заданные пользователем правила семантики;
- схема DSRL содержит спецификации принятого по умолчанию содержимого.

# *7. Проверка содержимого NETCONF*

В этом разделе описано, как схемы, созданные при отображении YANG-DSDL представляются для проверки экземпляров документов XML, таких как содержимое хранилищ данных или сообщения NETCONF. Этапы проверки показаны на рисунке 2.

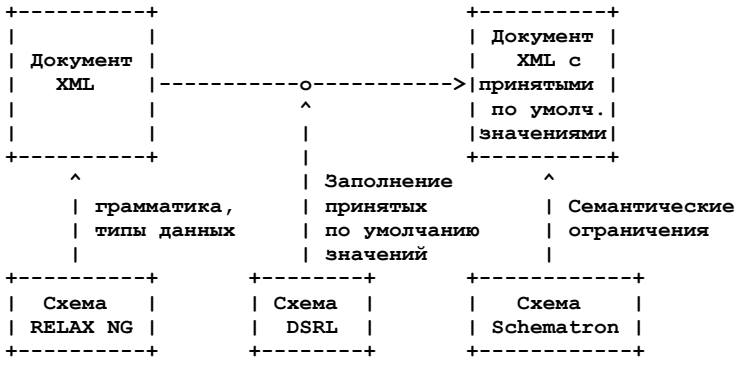


*Рисунок 2. Процедура проверки.*

1. Экземпляр документа XML проверяется в плане грамматики и типов данных по схеме RELAX NG.
2. Применяются заданные по умолчанию значения листьев и при необходимости добавляются их контейнеры-предки. Важно добавить неявные узлы до следующей проверки, поскольку спецификация YANG [RFC6020] требует, чтобы в используемом при оценке выражения XPath были указаны все принятые по умолчанию значения. Отметим, что этот шаг меняет набор информации в проверяемом документе XML.
3. Проверяются семантические ограничения с использованием схемы Schematron.

# *8. Устройство*

Модели данных YANG можно отобразить в схемы DSDL множеством способов. Описанная в документе процедура отображения использует несколько конкретных решений, описанных в последующих параграфах.

## 8.1. Гибридная схема

Как указано в разделе 6, первый этап сопоставления создаёт промежуточный документ - гибридную схему, задающую все ограничения для модели данных с использованием синтаксиса RELAX NG и дополнительных аннотаций. Её нельзя напрямую использовать для проверки и на деле это даже не будет действительной схемой RELAX NG, поскольку в документе будет несколько схем, разделённых элементами аннотаций.

Каждому входному модулю YANG соответствует один вложенный блок (grammar) гибридной схемы. Такое разделение входных модулей YANG позволяет в каждый вложенный блок включить определения именованных шаблонов со своим пространством имён, чтo важно для отображения группировок YANG (см. 9.2. Отображение группировок и определений типов YANG).

В дополнение к ограничениям грамматики и типов данных модули YANG предоставляют другие важные сведения, которые нельзя выразить в схеме RELAX NG - семантические ограничения, принятые по умолчанию значения, метаданные, документацию и т. п. Такая информация представляется в гибридной схеме атрибутами и элементами XML, относящимися к пространству имён с URI urn:ietf:params:xml:ns:netmod:dsdl-annotations:1. Полный список таких аннотаций приведён в параграфе 5.3, а правила их применения описаны ниже.

Модули YANG задают модели не только для данных конфигурации и состояния, но и для (множества) операций RPC [RFC4741] и/или уведомлений [RFC5277]. Чтобы зафиксировать все эти типы данных в одном документе схемы, в гибридной схеме применяются специальные маркеры, заключающие в себе субсхемы для данных конфигурации и состояния, отдельные операции RPC (вход и выход) и отдельные уведомления. В маркерах применяются указанные в таблице 3 элементы XML в пространстве имён связанных с NETMOD аннотаций (URI urn:ietf:params:xml:ns:netmod:dsdl-annotations:1).

*Таблица 3. Элементы маркеров в гибридной схеме.*

| ***Имя элемента*** | ***Охват*** |
|---|---|
| nma:data | Данные конфигурации и состояния |
| nma:rpcs | Все операции RPC |
| nma:rpc | Отдельная операция RPC |
| nma:input | Запрос RPC |
| nma:output | Отклик RPC |
| nma:notifications | Все уведомления |
| nma:notification | Отдельное уведомление |

В качестве примера рассмотрим модель с двумя модулями YANG example-a и example-b, которые определяют узлы данных в пространствах имён http://example.com/ns/example-a и http://example.com/ns/example-b. Модуль example-a задаёт данные конфигурации и состояния, методы RPC и уведомления, а example-b - только данные конфигурации и состояния. Гибридную схему можно вредставить в виде

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:nma="urn:ietf:params:xml:ns:netmod:dsdl-annotations:1"
         xmlns:exa="http://example.com/ns/example-a"
         xmlns:exb="http://example.com/ns/example-b"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
  <start>
    <grammar nma:module="example-a"
             ns="http://example.com/ns/example-a">
      <start>
        <nma:data>
          ... данные конфигурации и состояния из example-a...
        </nma:data>
        <nma:rpcs>
          <nma:rpc>
            <nma:input>
              <element name="exa:myrpc">
                ...
              </element>
            </nma:input>
            <nma:output>
              ...
            </nma:output>
          </nma:rpc>
          ...
        </nma:rpcs>
        <nma:notifications>
          <nma:notification>
            <element name="exa:mynotif">
              ...
            </element>
          </nma:notification>
          ...
        </nma:notifications>
      </start>
      ... локальные определения именованных шаблонов из example-a...
    </grammar>
    <grammar nma:module="example-b"
             ns="http://example.com/ns/example-a">
      <start>
        <nma:data>
          ... данные конфигурации и состояния из example-b...
        </nma:data>
        <nma:rpcs/>
        <nma:notifications/>
      </start>
      ... локальные определения именованных шаблонов из example-b...
    </grammar>
  </start>
  ... глобальные определения именованных шаблонов ...
</grammar>
```

Полная гибридная схема для модели данных сервера DHCP приведена в Приложении C.2.

## 8.2. Модульность

YANG и RELAX NG предоставляют средства для обеспечения модульности, т. е. разделения содержимого полной схемы на отдельные модули и комбинирования или использования их разными способами. Однако подход в YANG и RELAX NG различается. Модульность RELAX NG подходит для специализированных комбинаций небольшого числа схем, а YANG предполагает большой набор модулей, подобно SNMP MIB. Ниболее важные различия указаны ниже.

- В YANG модуль A, импортирующий модуль B, получает доступ к определениям (grouping и typedef) на верхнем уровне модуля B. Однако дерево данных модуля B не импортируется с определениями. В RELAX NG шаблон <rng:include> импортирует определения именованных шаблонов и всё дерево включённой схемы.

- Имена импортированных группировок и определений типов YANG указываются с пространством имён импортированного модуля, а имена узлов данных, содержащихся в импортированных группировках, при использовании в импортирующем модуле становятся частью его пространства имён. В RELAX NG имена шаблонов не уточняются, а именованные шаблоны в импортирующем и импортируемом шаблоне используют общее плоское пространство имён. Содержимое именованных шаблонов RELAX NG может сохранять пространство имён схемы, где они определены, или наследовать пространство имён импортирующего модуля, как в YANG. Однако для второго варианта определения именованных шаблонов должны включаться из внешней схемы, которая должна быть специально подготовлена (см. главу 11 в [Vli04]).

Для максимально возможного отображения модульности YANG в RELAX NG проверяющую схему RELAX NG (результат второго этапа отображения) нужно разделить на два файла, один из которых содержит все глобальные определения, отображённые из группировок YANG верхнего уровня всех входных модулей YANG. Этой схеме RELAX NG **недопустимо** определять какое-либо пространство имён через атрибут @ns. Второй файл схемы RELAX NG определяет фактические деревья данных, отображённые из входных модулей YANG и каждое из них помещается в свой блок (grammar). Вложенные блоки, в которых используется хотя бы одно глобальное определение, **должны** включать первую схему с определениями, а также **должны** задать локальное пространство имён с помощью атрибута @ns. Таким способом можно использовать глобальные определения внутри разных встроенных блоков (grammar), каждый раз принимая своё локальное пространство имён.

Определения именованных шаблонов, отображённые из группировок YANG, не относящихся к верхнему уровню, **должны** помещаться внутрь встроенного блока (grammar), соответствующего модулю YANG, где группировка задана.

В гибридной схеме нужно различать глобальные и неглобальные определения именованных шаблонов, сохраняя при этом гибридную схему в одном файле. Это достигается, как указано ниже.

- Каждое глобальное определение **должно** размещаться как потомок внешнего элемента <rng:grammar> (корневой документ гибридной схемы).

- Каждое неглобальное определение **должно** размещаться как потомок соответствующего вложенного элемента <rng:grammar>.

YANG также позволяет разделить модуль на несколько субмодулей. Однако, такая модульность не влияет на отображение, поскольку субмодули не меняют область действия идентификаторов и пространство имён. Содержимое субмодулей обрабатывается, как будто текст субмодуля размещён непосредственно в модуле.

## 8.3. Детализация

RELAX NG поддерживает разные стили структурирования схем. Например, один из крайних случаей называется русской матрешкой (Russian Doll) и задаёт структуру экземпляра документа XML в одной иерархии. Другой крайний случай (плоский стиль) использует подход, похожий на язык схем определения типа данных (Data Type Definition или DTD), где каждый элемент XML соответствует определению именованного шаблона. На практике обычно применяется что-то среднее между этими вариантами.

YANG в принципе поддерживает оба стиля, но в большинстве случаев модули организуются подобно матрешке, что позволяет лучше увидеть структуру данных конфигурации. Группировки обычно определяются лишь для содержимого, предназначенного для многократного использования в разных местах с помощью операторов uses. Схемы RELAX NG обычно более плоские, поскольку в RELAX NG нужна более тонкая детализация для расширяемости схем - можно заменять или изменять лишь фрагменты схемы, которые выделены как именованные шаблоны. Для YANG это не проблема, поскольку операторы augment и refine можно углублять с помошью выражений для путей до произвольной глубины имеющихся структур.

В общем случае невозможно сопоставить мощные механизмы расширения YANG с доуступными в RELAX NG. По этой причине отображение по существу сохраняет детализацию исходной модели данных YANG - группировки и определения производных типов обычно имеют прямые аналоги в определениях именованных шаблонов результирующей схемы RELAX NG.

## 8.4. Обработка пространств имён XML

Большинство современных языков схем XML, включая RELAX NG, Schematron и DSRL, поддерживают схемы для составных (compound) документов XML с элеменатми из разных пространство имён. Это полезно для наших целей, поскольку отображение YANG-DSDL поддерживает на входе несколько модулей YANG, что естественно ведёт в составным схемам документов. В RELAX NG есть два вариант определения целевых пространств имён в схемах:

1. традиционный способ XML через атрибут @xmlns:xxx;

2. один из URI целевых пространств имён может быть объявлен через атрибуто @ns.

В гибридной схеме и схемах проверки RELAX NG, создаваемых на втором этапе, пространства имён должны объявляться, как показано ниже.

1. Корень <rng:grammar> **должен** иметь атрибуты @xmlns:xxx, объявляющие префиксы всех используемых в модели пространств имён. Префиксам **следует** быть идентичными заданным в операторах prefix. Реализация отображения **должна** разрешать все конфликты префиксов из разных входных модулей при их возникновении.

2. Каждый вложенный элемент <rng:grammar> **должен** объявлять пространство имён соответствующего модуля с помощью атрибута @ns. Таким образом, имена узлов, заданные глобальными именованными шаблонами, могут принимать локальные пространства имён каждого встроенного блока (grammar), как указано в параграфе 8.2.

Это проиллюстрировано примером в конце параграфа 8.1.

Схемы DSRL могут объявлять любой число целевых пространств имён через стандартные атрибуты XML xmlns:xxx. В Schematron требуется определять все используемые пространства имён в субэлементах <sch:ns> элемента <sch:schema>.

# *9. Отображение моделей данных YANG в гибридную схему*

В этом разделе разъясняются основные принципы первого этапа отображения, результатом которого является гибридная схема, описанная в параграфе 8.1. Подробные описания отображений операторов YANG даны в разделе 10.

## 9.1. Правила вхождения для узлов данных

В языках схем DSDL ограничения для размещения узла всегда размещаются вместе с этим узлом. В схеме RELAX NG, например, шаблон <rng:optional> служит родительским элементом для шаблона, определяющего лист или иной элемент. DSRL задаёт принятое по умолчанию содержимое раздельно для каждого узла, независимо от того, является ли он узлом. Для листьев в модулях YANG ограничения вхождения легко выводится из субоператоров leaf. Однако для контейнеров YANG часто требуется проверить всю ветвь для определения ограничений на вхождение контейнера.

Поэтому одной из целей первого этапа отображения является вывод ограничений на вхождение для всех узлов данных и подобающая маркировка соответствующих шаблонов <rng:element> в гибридной схеме, чтобы любая процедура на втором этапе преобразования могла использовать эти сведения и не требовалось заново проверять ветвь.

Сначала нужно решить, должен ли этот узел данных всегда присутствовать в допустимой конфигурации. Если так, узел называется обязательным, в ином случае - необязательным. Это ограничение тесно связано с понятием обязательных узлов в параграфе 3.1 [RFC6020]. Единственное отличие состоит в том, что этот документ считает обязательными ключи списков.

Другое ограничение вхождения связано с семантикой default и возможностью удаления пустых контейнеров отсутствия (non-presence). В результате набор сведений допустимой конфигурации может быть изменён путём добавления или удаления некоторых элементов leaf или container без изменения смысла конфигурации. В этом документе такие элементы называются неявными. В гибридной схеме они могут быть указаны шаблоном RELAX NG, начинающимся с атрибута @nma:default или @nma:implicit.

Отметим, что оба ограничения на вхождение применимы к контейнерам на верхнем уровне дерева данных, а также к другим контейнерам при условии, что их родительский узел имеется в экземпляре документа. В качестве примера рассмотрим приведённый ниже фрагмент YANG.

```
container outer {
    presence 'Наличие outer что-то означает.';
    container c1 {
        leaf foo {
            type uint8;
            default 1;
        }
    }
    container c2 {
        leaf-list bar {
            type uint8;
            min-elements 0;
        }
    }
    container c3 {
        leaf baz {
            type uint8;
            mandatory true;
        }
    }
}
```

Здесь контейнер outer включает субоператор presence, который говорит, что контейнер является необязательным и не неявным. Если outer нет в конфигурации, не будет и его дочерних контейнеров. Однако при наличии outer имеет смысл спросить, какой из контейнеров-потомков является необязательным, а какой - неявным. В примере c1 является необязательным и неявным, c2 - необязательным но не неявным, c3 - обязательным (следовательно, не неявным).

В последующих параграфах приведены точные правил для определения, является контейнер обязательным или необязательным и является ли он неявным. Для упрощения рекурсивного определения этих характеристик вхождения полезно задать их и для других типов узлов схемы YANG - leaf, list, leaf-list, anyxml, choice.

### *9.1.1. Обязательные и необязательные узлы*

Ниже приведены правила, позволяющие классифицировать узел как обязательный или необязательный:

- узлы leaf, anyxml, choice являются обязательными, если они включают субоператор mandatory true, для узлов choice это означает, что должен существовать хотя бы 1 узел в одной ветви case;
- узел leaf является обязательным, если он объявлен ключом списка;
- узлы list и leaf-list обязательны при наличии субоператора min-elements со значением больше 0;

- узел container является обязательным, если его определение не включает субоператор presence и в контейнере имеется хотя бы один обязательный лист-потомок.

Остальные узлы считаются необязательными.

В RELAX NG определения необязательных узлов должны явно помещаться в элемент <rng:optional>. Отображение **должно** использовать приведённые выше правила для определения необязательных узлов YANG и при обнаружении таковых помещать элемент <rng:optional> в гибридную схему. Варианты в <rng:choice> **недопустимо** указывать в гибридной схеме как необязательные. Если choice в YANG является необязательным, **должен** применяться элемент <rng: optional> для включения всего шаблона <rng:choice>.

### *9.1.2. Неявные узлы*

Ниже приведены правила, позволяющие классифицировать узел как неявный:

- узлы list, leaf-list, anyxml никогда не бывают неявными;
- узел leaf является неявным, если он имеет заданное по умолчанию значение напрямую или через тип данных;
- узел container является неявным, если у него нет субоператора presence, ни один их потомков не является обязательным и хотя бы один потомок является неявным.

В гибридной схеме все неявные контейнеры, а также листья, которые получают заданное по умолчанию значение от typedef и не имеют атрибута @nma:default, **должны** помечаться атрибутом @nma:implicit со значением true.

Отметим, что в параграфе 7.9.3 [RFC6020] заданы иные правила, которые должны учитываться при решении вопроса, является ли неявным container или leaf, размещённый внутри case оператора choice. В частности, лист или контейнер в case может быть неявным лишь при условии, что case присутствует в аргументе оператора default для choice. Однако этого недостаточно и решение зависит от конкретного экземпляра документа XML, а именно от наличия или отсутствия узлов из других (не принятых по умолчанию) case (см. 11.3. Отображение принятых по умолчанию значений в DSRL).

## *9.2. Отображение группировок и определений типов YANG*

Группировки и определения типов YANG обычно отображаются в именованные шаблоны RELAX NG, однако есть несколько предостережений, которые нужно учитывать. Прежде всего, определения типов и группировки YANG могут присутствовать на всех уровнях иерархии модуля и с ними связана область лексической видимости (параграф 5.5 в [RFC6020]. Во-вторых, символы верхнего уровня из внешних модулей могут импортироваться как полные имена, представленные с префиксом пространства имён внешнего модуля. Именованные шаблоны в RELAX NG (локальные и импортированные через шаблон <rng:include>) используют одно пространство имён, а внутри блока (grammar) они всегда глобальны - их определения могут присутствовать лишь на верхнем уровне как потомки элемента <rng:grammar>. Следовательно, всякий раз при отображении группировок и определений типов YANG в определения именованных шаблонов RELAX NG в их именах **должна** устраняться неоднозначность, чтобы избежать конфликтов. При отображении применяется описанная ниже процедура изменения имён группировок и определений типов.

- К именам grouping и typedef с верхнего уровня иерархии модуля YANG добавляется префикс из имени модуля и 2 символов подчёркивания (__).
- К остальным именам grouping и typedef (не присутствующим на верхнем уровне модуля YANG) добавляется префикс из имени модуля, 2 символов подчёркивания (__) и всех узлов-предков, разделённых 2 символами подчёркивания.
- Поскольку имена groupings и typedef в YANG относятся к разным пространствам имён, в начале имён всех группировок добавляется 1 символ подчёркивания.

Дополнительная сложность связана с правилами YANG для упорядочения субэлементов (см., например, параграф 7.5.7 в [RFC6020]). Во входных и выходных параметрах RPC субэлементы должны размещаться в порядке, заданном моделью данных, остальные субэлементы могут иметь произвольный порядок. Поэтому при использовании группировки во входных или выходных параметрах RPC и ещё где-нибудь, она **должна** отображаться в 2 разных определения именованных шаблонов - один с фиксированным порядком, другой - с произвольным. Чтобы различать их, для фиксированного порядка **должен** добавляться суффикс __rpc.

Рассмотрим в качестве пример модуль YANG, импортирующий стандартный модуль ietf-inet-types [RFC6021].

```
module example1 {
    namespace "http://example.com/ns/example1";
    prefix ex1;
    typedef vowels {
        type string {
            pattern "[aeiouy]*";
        }
    }
    grouping "grp1" {
        leaf "void" {
            type "empty";
        }
    }
    container "cont" {
        leaf foo {
            type vowels;
        }
        uses "grp1";
    }
}
```

Гибридная схема, созданная на первом этапе отображения, будет содержать 2 (глобальных) определения именованных шаблонов, как показано ниже.

```
<rng:define name="example1__vowels">
  <rng:data type="string">
    <rng:param name="pattern">[aeiouy]*</rng:param>
  </rng:data>
</rng:define>

<rng:define name="_example1__grp1">
  <rng:optional>
    <rng:element name="void">
      <rng:empty/>
    </rng:element>
  </rng:optional>
</rng:define>
```

### *9.2.1. Уточнения и дополнения YANG*

Группировки YANG представляют ту же концепцию, что и определения именованных шаблонов в RELAX NG и оба языка предоставляют механизмы для их последующего изменения. Однако в RELAX NG изменяются сами определения, а YANG использует субоператоры uses, меняющие расширения группировок.

- Оператор refine позволяет изменить параметры узла схемы внутри группировки, указанной родительским оператором uses.
- Оператор augment позволяет добавить в группировку новые узлы схемы.

Операторы refine и augment имеют достаточно эффективные возможности, используя выражения в стиле XPath как свои аргументы для обращения к узлам схемы, расположенным сколь угодно глубоко внутри содержимого группировки. Изменения определений именованных шаблонов в RELAX NG применяются исключительно на верхнем уровне содержимого именованного шаблона. Для достижения изменяемости именованных шаблонов, сравнимой с YANG, схема RELAX NG должна быть совершенно плоской (8.4. Обработка пространств имён XML) и станет неудобочитаемой.

Поскольку цель описываемого здесь отображения заключается в генерации специализированных схем DSDL, принято решение избегать этих сложностей и преобразовывать grouping, refine и/или augment «на месте». Иными словами, каждый оператор uses с субоператорами refine и/или augment заменяется содержимым соответствующей группировки, применяются изменения, заданные операторами refine и augment, после чего полученный фрагмент схемы YANG отображается, как будто опосредованности uses/grouping не было.

Если в содержимом группировки имеются дополнительные операторы uses, для них тоже может потребоваться преобразование. Это необходимо, если имеющиеся пары uses и grouping находятся на «пути изменения», заданном аргументом оператора refine или augment.

Рассмотрим в качестве примера приведённый ниже модуль YANG.

```
module example2 {
    namespace "http://example.com/ns/example2";
    prefix ex2;
    grouping leaves {
        uses fr;
        uses es;
    }
    grouping fr {
        leaf feuille {
            type string;
        }
    }
    grouping es {
        leaf hoja {
            type string;
        }
    }
    uses leaves;
}
```

Полученная в результате гибридная схема содержит три глобальных определения именованных шаблонов для трёх группировок.

```
<rng:define name="_example2__leaves">
  <rng:interleave>
    <rng:ref name="_example2__fr"/>
    <rng:ref name="_example2__es"/>
  </rng:interleave>
</rng:define>

<rng:define name="_example2__fr">
  <rng:optional>
    <rng:element name="feuille">
      <rng:data type="string"/>
    </rng:element>
  </rng:optional>
</rng:define>

<rng:define name="_example2__es">
  <rng:optional>
    <rng:element name="hoja">
      <rng:data type="string"/>
    </rng:element>
  </rng:optional>
</rng:define>
```

Связанная с данными конфигурации часть гибридной схемы будет одной ссылкой на именованный шаблон.

```
<nma:data>
  <rng:ref name="_example2__leaves"/>
</nma:data>
```

Предположим, что оператор uses leaves включает субоператор refine, например,

```
uses leaves {
    refine "hoja" {
        default "alamo";
    }
}
```

Результирующая гибридная схема теперь содержит лишь одно определение именованного шаблона - _example2__fr. Две другие группировки leaves и es должны быть преобразованы, поскольку они находятся на «пути изменения», т. е. содержат уточнение hoja. Связанная с данными конфигурации часть гибридной схемы будет иметь вид

```
<nma:data>
  <rng:interleave>
    <rng:ref name="_example2__fr"/>
    <rng:optional>
      <rng:element name="ex2:hoja" nma:default="alamo">
        <rng:data type="string"/>
      </rng:element>
    </rng:optional>
  </rng:interleave>
</nma:data>
```

### *9.2.2. Цепочки производных типов*

В RELAX NG нет эквивалента созданию производных типов YANG, позволяющего ограничить встроенный тип (возможно, за несколько шагов) путём добавления ограничений. Всякий раз, когда производный тип YANG применяется без ограничений (как субоператор leaf или другого typedef), оператор type просто отображается в ссылку на именованный шаблон <rng:ref>, а определение типа - в определение именованного шаблона RELAX NG <rng:define>. Однако при указании ограничений субоператорами оператора type определение типа **должно** преобразовываться в этой точке, чтобы в гибридной схеме отображался лишь встроенный тип предка, ограниченный поскольку лишь лишь «гранями», соответствующими сочетанию всех ограничений, найденный в цепочке производного типа и операторе type.

Рассмотрим в качестве примера модуль YANG, представленный ниже.

```
module example3 {
    namespace "http://example.com/ns/example3";
    prefix ex3;
    typedef dozen {
        type uint8 {
            range 1..12;
        }
    }
    leaf month {
        type dozen;
    }
}
```

Оператор type в листе month не имеет ограничений и поэтому отображается просто в ссылку <rng:ref name="example3__dozen"/>, а соответствующий именованный шаблон имеет вид

```
<rng:define name="example3__dozen">
  <rng:data type="unsignedByte">
    <rng:param name="minInclusive">1</rng:param>
    <rng:param name="maxInclusive">12</rng:param>
  </rng:data>
</rng:define>
```

Предположим, что определение листа month изменено, как показано ниже

```
leaf month {
    type dozen {
        range 7..max;
    }
}
```

Выходная схема RELAX NG не будет включать определения именованного шаблона и лист month будет отображён в

```
<rng:element name="ex3:month">
  <rng:data type="unsignedByte">
    <rng:param name="minInclusive">7</rng:param>
    <rng:param name="maxInclusive">12</rng:param>
  </rng:data>
</rng:element>
```

Отображение цепочек производных типов может дополнительно осложняться наличием оператора default в определениях типа. В простом случае, когда определение типа с оператором default применяется без ограничений, default отображается в атрибут @nma:default элемента <rng:define>. Однако при расширении определения типа в результате ограничений атрибут @nma:default, возникающий из расширенного типа или родительских типов в цепочке создания производного типа, должен присоединяться к шаблону, где выполняется преобразование. При наличии нескольких операторов default в цепочке производного типа применяется лишь ближайший к преобразуемому типу оператор default.

Рассмотрим вариант предыдущего примера.

```
module example3bis {
    namespace "http://example.com/ns/example3bis";
```

```
        prefix ex3bis;
        typedef dozen {
            type uint8 {
                range 1..12;
            }
            default 7;
        }
        leaf month {
            type dozen;
        }
    }
```

Оператор typedef в этом модуле отображается в приведённое ниже определение именованного шаблона.

```
    <rng:define name="example3bis__dozen" @nma:default="7">
      <rng:data type="unsignedByte">
        <rng:param name="minInclusive">1</rng:param>
        <rng:param name="maxInclusive">12</rng:param>
      </rng:data>
    </rng:define>
```

Если тип dozen ограничен при использовании в определении листа month (как в предыдущем примере), тип dozen преобразуется и @nma:default становится атрибутом определения элемента <ex3bis:month>.

```
    <rng:element name="ex3bis:month" @nma:default="7">
      <rng:data type="unsignedByte">
        <rng:param name="minInclusive">7</rng:param>
        <rng:param name="maxInclusive">12</rng:param>
      </rng:data>
    </rng:element>
```

Однако при наличии в листе month субоператора default, заданное по умолчанию значение типа dozen игнорируется.

## 9.3. Трансляция выражений XPath

YANG использует полный синтаксис XPath 1.0 [XPath] для аргументов must, when, path. Поскольку имена узлов данных в модуле YANG всегда относятся к пространству имён этого модуля, в YANG принято упрощение, похожее на концепцию принятого по умолчанию пространства имён XPath 2.0 - в именах узлов в выражениях XPath не требуется указывать префикс пространства имён внутри модуля, где эти узлы определены (предполагается локальное пространство). Поэтому все выражения XPath **должны** транслироваться в полном соответствии с XPath 1.0 - к каждому узлу без префикса **должен** добавляться в начале префикс пространства имён локального модуля, заданный оператором prefix.

Выражения XPath внутри группировок верхнего уровня требуют особого внимания, поскольку имена содержащихся в них узлов без префиксов должны принимать пространство имён модуля, где применяется группировка (8.2. Модульность). Для этого локальный префикс **должен** быть представлен в гибридной схеме с использованием переменной $pref. В Schematron для таких выражений XPath подставляется соответствующее значение переменной через параметр в абстрактном шаблоне, в который отображается группировка YANG (11.2. Отображение семантических ограничений в Schematron). Например, выражение XPath /dhcp/max-lease-time из модуля YANG с префиксом dhcp будет транслироваться в

- $pref:dhcp/$pref:max-lease-time, если оно находится в группировке верхнего уровня;
- dhcp:dhcp/dhcp:max-lease-time в ином случае.

В YANG также применяются другие выражения в стиле XPath - идентификаторы ключей и идентификаторы узлов-потомков схемы (см. ABNF для descendant-schema-nodeid в параграфе 12 [RFC6020]). Эти выражения **должны** транслироваться с добавлением префикса локального модуля.

## 9.4. Расширения языка YANG

YANG позволяет расширять себя, добавляя новые операторы с ключевыми словами из специальных пространств имён. Такие расширения должны сначала объявляться с помощью оператора extension, затем их можно применять как стандартные операторы YANG, от которых они отличаются префиксом пространства имён, определяющим ключевое слово расширения. В RELAX NG имеется похожий механизм расширения - элементы и атрибуты XML с именами из внешних пространств имён могут помещаться почти в любое место схемы RELAX NG.

Расширения YANG могут (но не обязаны) иметь значение в контексте схем DSDL, поэтому реализация **может** игнорировать любое или все расширения. Если расширение не игнорируется, оно **должно** отображаться в элементы и/или атрибуты XML, которые точно соответствуют YIN-форме расширения (см. параграф 11.1 в [RFC6020]).

Рассмотрим в качестве примера расширение, заданное модулем acme.

```
    extension documentation-flag {
        argument number;
    }
```

Это расширение можно использовать в этом же или ином модуле, например, как показано ниже.

```
    leaf folio {
        acme:documentation-flag 42;
        type string;
    }
```

Если это расширение учитывается при отображении, последнее будет иметь вид

```
    <rng:element name="acme:folio">
       <acme:documentation-flag number="42"/>
       <rng:data type="string"/>
    </rng:element>
```

Отметим, что сам оператор extension не отображается.

# 10. Отображение операторов YANG в гибридную схему

Каждый параграф этого раздела посвящён одному оператору YANG и описывает его отображение в гибридную схему. Параграфы отсортированы по алфавиту в соответствии с ключевыми словами операторов.

Каждый оператор YANG отображается во фрагмент XML, обычно с одним элементом и атрибутом, но возможны и более крупные структуры. Процедура отображения рекурсивна по природе, что означает продолжение отображений субоператоров после отображения самого оператора. Один элемент результирующего отображения становится родителем других фрагментов, полученных при отображении субоператоров. Любые отклонения от такой рекурсивной обработки указываются явно.

Правила кодирования YANG XML транслируются в приведённые ниже правила упорядочения субэлементов.

1. Внутри ветви <nma:rpcs> (параметры ввода и вывода для операции RPC) порядок субэлементов фиксирован и их определения в гибридной схеме **должны** следовать порядку в исходном модуле YANG.
2. При отображении оператора list все ключи **должны** размещаться впереди других субэлементов с сохранением порядка, заданного в операторе key. Порядок остальных субэлементов (не ключей) не задаётся, поэтому определения в гибридной схеме **должны** помещаться в элемент <rng:interleave>.
3. В иных случаях порядок субэлементов может быть произвольным, поэтому определения в гибридной схеме **должны** помещаться в элемент <rng:interleave>.

В этом разделе применяются два соглашения, указанных ниже.

- Аргумент отображаемого оператора обозначается словом ARGUMENT.
- Элемент схемы RELAX NG, становящийся родителем в полученном фрагменте XML, указывается словом PARENT.

## 10.1. Оператор anyxml

Этот оператор отображается в элемент <rng:element> и ARGUMENT с добавленным впереди префиксом локального пространства имён становится значением атрибута @name. Содержимое элемента <rng:element> имеет вид

```
<rng:ref name="__anyxml__"/>
```

При наличии у anyxml субоператоров они **могут** отображаться в дочерние элементы <rng:element>.

Если в любом из входных модулей YANG имеется хотя бы один оператор anyxml, **должно** добавляться в точности 1 определение шаблона в схему RELAX NG как потомок корневого элемента <rng:grammar> (см. стр 172 в [Vli04]).

```
<rng:define name="__anyxml__">
  <rng:zeroOrMore>
    <rng:choice>
      <rng:attribute>
        <rng:anyName/>
      </rng:attribute>
      <rng:element>
        <rng:anyName/>
        <rng:ref name="__anyxml__"/>
      </rng:element>
      <rng:text/>
    </rng:choice>
  </rng:zeroOrMore>
</rng:define>
```

Например, оператор YANG в модуле с пространством имён с префиксом yam

```
anyxml data {
    description "Произвольное содержимое XML.";
}
```

отображается в приведённый ниже фрагмент.

```
<rng:element name="yam:data">
    <a:documentation>Произвольное содержимое XML</a:documentation>
    <rng:ref name="__anyxml__"/>
</rng:element>
```

Узел anyxml является необязательным, если в нем нет субоператора mandatory true. В этом случае элемент <rng:element> **должен** помещаться в <rng:optional>, если anyxml не является потомком оператора choice, формируя таким образом сокращение case для этого выбора (см. 9.1.1. Обязательные и необязательные узлы).

## 10.2. Оператор argument

Этот оператор не отображается в выходную схему, но применяются правила обработки расширений из параграфа 9.4.

## 10.3. Оператор augment

Как и субоператор uses, этот оператор обрабатывается как часть отображения uses, описанного в параграфе 10.57.

На верхнем уровне модуля или субмодуля оператор augment служит для дополнения дерева схемы другого модуля YANG. Если расширяемый модуль не обрабатывается в том же отображении, оператор augment на верхнем уровне **должен** игнорироваться. В ином случае содержимое оператора добавляется к внешнему модулю с пространством имён модуля, где указан оператор augment.

## 10.4. Оператор base

Этот оператор игнорируется, будучи субоператором identity, и обрабатывается с типом identityref при использовании как субоператора определения данного типа (см. 10.53.6. Тип identityref).

## 10.5. Оператор belongs-to

Этот оператор не применяется, поскольку обработка субмодулей всегда инициируется из основного модуля (см. 10.24. Оператор include).

## 10.6. Оператор bit

Этот оператор обрабатывается внутри типа bits (см. 10.53.4. Тип bits).

## 10.7. Оператор case

Этот оператор отображается в элемент <rng:group> или <rng:interleave> в зависимости от принадлежности к операции RPC. Если аргумент братского (sibling) оператора default равен ARGUMENT, **должен** добавляться атрибут @nma:implicit со значением true к этому элементу <rng:group> или <rng:interleave>. Атрибут @nma:implicit **недопустимо** применять для узлов верхнего уровня в case, не используемом по умолчанию (см. параграф 7.9.3 в [RFC6020]).

## 10.8. Оператор choice

Этот оператор отображается в элемент <rng:choice>. Если choice включает субоператор mandatory true, к элементу <rng:choice> **должен** добавляться атрибут @nma:mandatory со значением ARGUMENT, что может потребовать дополнительной обработки (см. параграф 11.2.1. Ограничения для обязательного choice). При отсутствии mandatory true элемент <rng:choice> **должен** помещаться в <rng:optional>. Варианты <rng:choice>, отображённые из оператора case или сокращения case, **недопустимо** делять необязательными.

## 10.9. Оператор config

Этот оператор отображается в атрибут @nma:config, а ARGUMENT становится его значением.

## 10.10. Оператор contact

Этот оператор **не следует** использовать в отображении, поскольку с гибридной может сопоставляться несколько модулей YANG, созданных разными авторами. Гибридная схема содержит ссылки на все входные модули в элементах Dublin Core <dc:source> (см. 10.34. Оператор module). Полномочные сведения об авторах содержатся в модулях YANG.

## 10.11. Оператор container

Используя правила параграфа 9.1.1, алгоритм отображения **должен** определить, задаёт ли оператор необязательный контейнер и, если это так, вставить элемент <rng:optional> и сделать его новым PARENT. Контейнер, заданный этим оператором, отображается в элемент <rng:element>, который становится потомком PARENT и использует ARGUMENT с добавленным впереди префиксом локального пространства имён как значение атрибута @name.

Используя правила параграфа 9.1.2, алгоритм отображения **должен** определить, является ли контейнер неявным и, если это так, добавить атрибут @nma:implicit со значением true к элементу <rng:element>.

## 10.12. Оператор default

Будучи субоператором leaf, этот оператор отображается в атрибут @nma:default для PARENT, а ARGUMENT становится его значением. Как субоператор typedef, оператор default тоже отображается в атрибут @nma:default со значением ARGUMENT. Размещение этого атрибута зависит от того, нужно ли преобразовывать определение типа при использовании.

- Если определение типа не преобразуется, @nma:default становится атрибутом шаблона <rng:define>, являющегося результатом отображения родительского typedef.
- В ином случае @nma:default будет атрибутом шаблона-предка RELAX NG, где выполняется преобразование.

Детали и пример представлены в параграфе 9.2. Отображение группировок и определений типов YANG.

Будучи субоператором choice, оператор default указывает принятый по умолчанию вариант и обрабатывается внутри оператора case (см. 10.7. Оператор case). Если принятый по умолчанию вариант использует сокращённую нотацию, где оператор case опущен, атрибут @nma:implicit со значением true присоединяется к узлу, представляющему заданный по умолчанию вариант в сокращённой нотации, или **может** вставляться дополнительный элемент <rng:group>, к которому и присоединяется атрибут @nma:implicit. В последнем случае конечный результат будет как при наличии case для принятого по умолчанию варианта.

Рассмотрим в качестве пример оператор choice в модуле с пространством имён yam.

```
choice leaves {
    default feuille;
    leaf feuille { type empty; }
    leaf hoja { type empty; }
}
```

Он отображается напрямую в

```
<rng:choice>
  <rng:element name="yam:feuille" nma:implicit="true">
    <rng:empty/>
  </rng:element>
  <rng:element name="yam:hoja">
    <rng:empty/>
  </rng:element/>
</rng:choice>
```

или принятый по умолчанию вариант может быть помещён в дополнительный элемент <rng:group>

```
<rng:choice>
  <rng:group nma:implicit="true">
```

```
      <rng:element name="yam:feuille">
        <rng:empty/>
      </rng:element>
    </rng:group>
    <rng:element name="yam:hoja">
      <rng:empty/>
    </rng:element/>
  </rng:choice>
```

## 10.13. Оператор description

Этот оператор отображается в элемент совместимости DTD <a:documentation>, а ARGUMENT будет его текстом. Для получения корректно форматированного компактного синтаксиса RELAX NG этот элемент **следует** вставлять как первого потомка PARENT.

## 10.14. Оператор deviation

Этот оператор игнорируется, однако предполагается, что все отклонения известны заранее и соответствующий изменения уже внесены во входные модули YANG.

## 10.15. Оператор enum

Этот оператор отображается в элемент <rng:value>, а ARGUMENT будет его текстом. Все субоператоры, кроме status, игнорируются, поскольку элемент <rng:value> не может включать аннотаций (см. раздел 6 в [RNG]).

## 10.16. Оператор error-app-tag

Этот оператор игнорируется, если он не является субоператором must. В последнем случае оператор отображается в элемент <nma:error-app-tag> (см. 10.35. Оператор must).

## 10.17. Оператор error-message

Этот оператор игнорируется, если он не является субоператором must. В последнем случае оператор отображается в элемент <nma:error-app-tag> (см. 10.35. Оператор must).

## 10.18. Оператор extension

Этот оператор игнорируется, однако расширения языка YANG **могут** отображаться, как описано в параграфе 9.4.

## 10.19. Оператор feature

Этот оператор игнорируется.

## 10.20. Оператор grouping

Этот оператор отображается в определение именованного шаблона RELAX NG <rng:define>, но лишь при использовании заданной оператором группировки без уточнений и дополнений в каком-либо из входных модулей. В этом случае определение именованного шаблона становится потомком элемента <rng:grammar>, а его именем будет значение ARGUMENT, изменённое в соответствии с правилами параграфа 9.2. Отображение группировок и определений типов YANG.

Как указано в параграфе 8.2. Модульность, определение именованного шаблона должно размещаться как потомок:

- элемента <rng:grammar>, если группировка задана на верхнем уровне входного модуля YANG;
- вложенного элемента <rng:grammar>, соответствующего модулю, где задана группировка, в ином случае.

При использовании группировки с уточнением или дополнением, она преобразуется так, чтобы применить уточнения и дополнения для предписанных узлов схемы на месте (см. 9.2.1. Уточнения и дополнения YANG).

Реализация **может** предлагать вариант отображения всех операторов grouping в определения именованных шаблонов в выходной схеме RELAX NG, даже если на них нет ссылок. Это полезно при отображении библиотечных модулей YANG, обычно содержащих лишь typedef и grouping.

## 10.21. Оператор identity

Этот оператор отображается в показанное ниже определение именованного шаблона, размещаемое как потомок корневого элемента <rng:grammar>.

```
  <rng:define name="__PREFIX_ARGUMENT">
    <rng:choice>
      <rng:value type="QName">PREFIX:ARGUMENT</rng:value>
      <rng:ref name="IDENTITY1"/>
      ...
    </rng:choice>
  </rng:define>
```

где

PREFIX - префикс, используемый в гибридной схеме для пространства имён модуля с определением identity;

IDENTITY1 - имя шаблона, соответствующего идентификатору, выведенному из текущего identity. Для каждого такого идентификатора **должен** присутствовать в точности 1 элемент <rng:ref>.

Рассмотрим пример из параграфа 7.16.3 в ([RFC6020], где заданы 2 identity в двух входных модулях YANG.

```
  module crypto-base {
    namespace "http://example.com/crypto-base";
```

```
    prefix "crypto";
    identity crypto-alg {
      description
        "Базовый идентификатор для вывода всех криптоалгоритмов.";
      }
  }

  module des {
    namespace "http://example.com/des";
    prefix "des";
    import "crypto-base" {
      prefix "crypto";
    }
    identity des {
      base "crypto:crypto-alg";
      description "Алгоритм DES";
    }
    identity des3 {
      base "crypto:crypto-alg";
      description "Алгоритм Triple DES";
    }
  }
```

Идентификаторы будут отображаться в приведённые ниже определения именованных шаблонов.

```
  <define name="__crypto_crypto-alg">
    <choice>
      <value type="QName">crypto:crypto-alg</value>
      <ref name="__des_des"/>
      <ref name="__des_des3"/>
    </choice>
  </define>
  <define name="__des_des">
    <value type="QName">des:des</value>
  </define>
  <define name="__des_des3">
    <value type="QName">des:des3</value>
  </define>
```

## 10.22. Оператор if-feature

ARGUMENT вместе с аргументами всех братских (sibling) операторов if-feature (с добавленными префиксами при их отсутствии) **должны** собираться в разделённый пробелами список, становящийся значением атрибута @nma:if-feature, который присоединяется к PARENT.

## 10.23. Оператор import

Для этого оператора нет специального отображения. Модуль, имя которого указано в ARGUMENT, анализируется, чтобы импортирующий модуль мог использовать его группировки, определения типов и идентификаторы верхнего уровня, а также дополнения дерева данных импортируемого модуля. Если оператор import имеет субоператор revision, **должен** использоваться соответствующий выпуск модуля, механизм поиска которого выходит за рамки документа.

## 10.24. Оператор include

Для этого оператора нет специального отображения. Субмодуль, имя которого указано в ARGUMENT, анализируется и его содержимое отображается как будто оно приведено непосредственно в тексте основного модуля. Если оператор include имеет субоператор revision, **должен** использоваться соответствующий выпуск модуля, механизм поиска которого выходит за рамки документа.

## 10.25. Оператор input

Этот оператор обрабатывается в рамках rpc, см. 10.50. Оператор rpc.

## 10.26. Оператор key

Этот оператор отображается в атрибут @nma:key. Значение ARGUMENT **должно** транслироваться так, что к каждому ключу добавлялся префикс пространства имён локального модуля. Результат трансляции становится значением атрибута @nma:key.

## 10.27. Оператор leaf

Этот оператор отображается в элемент <rng:element>, ARGUMENT с префиксом локального пространства имён становится значением атрибута @name. Если лист не является обязательным (не включает mandatory true), и не указан среди ключей содержащего его списка, элемент <rng:element> **должен** помещаться в <rng:optional>, за исключением случая, когда leaf является потомком choice и, таким образом, представляет сокращение case для этого выбора (см. 9.1.1. Обязательные и необязательные узлы).

## 10.28. Оператор leaf-list

Этот оператор отображается в блок, помещённый в элемент <rng:zeroOrMore> или <rng:oneOrMore> в зависимости от значения субоператора min-elements (0 для первого случая, применяемого по умолчанию, положительное значение для второго). Элемент <rng:zeroOrMore> или <rng: oneOrMore> становится родителем (PARENT). Затем добавляется элемент <rng:element> как потомок PARENT, а ARGUMENT с префиксом локального пространства имён становится значением атрибута @name. К элементу <rng:element> **должен** также добавляться атрибут @nma:leaf-list со значением true. Если leaf-list имеет субоператор min-elements с аргументом больше 1, к <rng:element> добавляется атрибут

@nma:min-elements со значением аргумента min-elements. При наличии субоператора max-elements со значением, отличным от unbounded добавляется атрибут @nma:max-elements со значением аргумента max-elements.

Рассмотрим в качестве примера leaf-list из модуля с префиксом пространства имён yam.

```
leaf-list foliage {
    min-elements 3;
    max-elements 6378;
    ordered-by user;
    type string;
}
```

Он отображается во фрагмент RELAX NG

```
<rng:oneOrMore>
  <rng:element name="yam:foliage" nma:leaf-list="true"
               nma:ordered-by="user"
               nma:min-elements="3" nma:max-elements="6378">
    <rng:data type="string"/>
  </rng:element>
</rng:oneOrMore>
```

## 10.29. Оператор length

Этот оператор обрабатывается в рамках string, см. 10.53.10. Тип string

## 10.30. Оператор list

Этот оператор отображается как и leaf-list (см. параграф 10.28). Единственным отличием является то, что наличие аннотации @nma:leaf-list **недопустимо** или атрибут **должен** иметь значение false.

При отображении субоператоров list порядок потомков элемента списка должен быть таким, чтобы ключи (при наличии) всегда указывались в порядке, заданном оператором key, и до остальных потомков (см. параграф 7.8.5 в [RFC6020]). В частности, если ключ списка задан в группировке, а сам узел списка не является частью этой группировки и положение оператора uses будет нарушать приведённые выше требования к порядку, группировка **должна** быть преобразована (uses заменяется содержимым группировки).

Рассмотрим в качестве примера фрагмент модуля YANG с префиксом yam.

```
grouping keygrp {
  leaf clef {
    type uint8;
  }
}
list foo {
  key clef;
  leaf bar {
    type string;
  }
  leaf baz {
    type string;
  }
  uses keygrp;
}
```

Он отображается в приведённый ниже фрагмент RELAX NG.

```
<rng:zeroOrMore>
  <rng:element name="yam:foo" nma:key="yam:clef">
    <rng:element name="yam:clef">
      <rng:data type="unsignedByte"/>
    </rng:element>
    <rng:interleave>
      <rng:element name="yam:bar">
        <rng:data type="string"/>
      </rng:element>
      <rng:element name="yam:baz">
        <rng:data type="string"/>
      </rng:element>
    </rng:interleave>
  </rng:element>
</rng:zeroOrMore>
```

Отметим, что группировка keygrp преобразована и определение yam:clef размещено перед шаблоном <rng:interleave>.

## 10.31. Оператор mandatory

Этот оператор может появляться как субоператор leaf, choice или anyxml. Если ARGUMENT имеет значение true, родительский узел отображается как обязательный (9.1.1. Обязательные и необязательные узлы).

В качестве субоператора choice этот оператор отображается в атрибут @nma:mandatory, добавляемый к PARENT. Значением этого атрибута служит аргумент родительского оператора choice.

## 10.32. Оператор max-elements

Этот оператор обрабатывается в рамках leaf-list или list, см. 10.28. Оператор leaf-list.

## 10.33. Оператор min-elements

Этот оператор обрабатывается в рамках leaf-list или list, см. 10.28. Оператор leaf-list.

## 10.34. Оператор module

Этот оператор отображается во встроенный шаблон <rng:grammar> с атрибутом @nma:module со значением ARGUMENT. Дополнительно **следует** создавать элемент <dc:source> как потомка этого <rng:grammar> со значением ARGUMENT как метаданные ссылки на входной модуль YANG (см. также 10.49. Оператор revision). Субоператоры module **должны** отображаться, как указано ниже.

- Субоператоры данных конфигурации и состояния отображаются в потомков элемента <nma:data>.
- Субоператоры содержимого запросов и откликов RPC отображаются в потомков элемента <nma:rpcs>.
- Субоператоры содержимого уведомлений отображаются в потомков элемента <nma:notifications>.

## 10.35. Оператор must

Этот оператор отображается в элемент <nma:must>, имеющий обязательный атрибут @assert (без пространства имён) с ARGUMENT, преобразованным в действительное выражение XPath (9.3. Трансляция выражений XPath). Элемент <nma:must> может иметь субэлементы, полученные из отображения субоператоров error-app-tag и error-message. Иные субоператоры must (description и reference) игнорируются.

Рассмотрим в качестве примера оператор YANG в модуле dhcp.

```
must 'current() <= ../max-lease-time' {
  error-message
    "Значение default-lease-time должно быть меньше max-lease-time";
}
```

Он отображается в

```
<nma:must assert="current()&lt;=../dhcp:max-lease-time">
  <nma:error-message>
    Значение default-lease-time должно быть меньше max-lease-time
  </nma:error-message>
</nma:must>
```

## 10.36. Оператор namespace

Этот оператор отображается двумя способами одновременно.

1. В атрибут @xmlns:PREFIX корневого элемента <rng:grammar>, где PREFIX - префикс пространства имён, заданный братским оператором prefix. ARGUMENT будет значением этого атрибута.
2. В атрибут @ns со значением ARGUMENT для PARENT, являющегося вложенным шаблоном <rng: grammar>.

## 10.37. Оператор notification

Этот оператор отображается в ветвь элемента <nma:notifications> в гибридной схеме, где PREFIX - префикс локального модуля YANG.

```
<nma:notification>
  <rng:element name="PREFIX:ARGUMENT">
    ...
  </rng:element>
</nma:notification>
```

Субоператоры notification отображаются в ветви <rng:element name="PREFIX:ARGUMENT">.

## 10.38. Оператор ordered-by

Этот оператор отображается в атрибут @nma:ordered-by со значением ARGUMENT (см. 10.28. Оператор leaf-list).

## 10.39. Оператор organization

Этот оператор игнорируется при отображении, поскольку гибридная схема может создаваться из нескольких модулей YANG, разработанных разными организациями. Гибридной схеме **следует** указывать все входные модули в элементах Dublin Core <dc:source> (см. 10.34. Оператор module). Сведения об организациях даны в исходных модулях YANG.

## 10.40. Оператор output

Этот оператор обрабатывается в рамках rpc, см. 10.50. Оператор rpc

## 10.41. Оператор path

Этот оператор обрабатывается в рамках типа leafref, см. 10.53.8. Тип leafref

## 10.42. Оператор pattern

Этот оператор обрабатывается в рамках типа string, см. 10.53.10. Тип string

## 10.43. Оператор position

Этот оператор игнорируется.

## 10.44. Оператор prefix

Этот оператор обрабатывается в рамках братского оператора namespace (10.36. Оператор namespace) или родительского оператора import (10.23. Оператор import). Как субоператор belongs-to (в субмодулях) оператор prefix игнорируется.

## 10.45. Оператор presence

Этот оператор влияет на отображение родительского контейнера (10.11. Оператор container), определение которого **должно** помещаться в <rng:optional>, независимо от содержимого (см. 9.1.1. Обязательные и необязательные узлы).

## 10.46. Оператор range

Этот оператор обрабатывается в рамках численных типов, см. 10.53.9. Численные типы.

## 10.47. Оператор reference

Этот оператор отображается в элемент <a:documentation>, а его текстом служит ARGUMENT с префиксом модуля.

## 10.48. Оператор require-instance

Этот оператор обрабатывается в рамках типа instance-identifier, см. 10.53.5. Типы enumeration и union.

## 10.49. Оператор revision

При отображении используется лишь последний экземпляр оператора revision (с наиболее поздней датой в ARGUMENT), который задаёт текущий выпуск входного модуля YANG [RFC6020]. Эту дату **следует** записать вместе с именем модуля YANG в соответствующий элемент Dublin Core <dc:source> (10.34. Оператор module), например, в виде <dc:source>YANG module 'foo', revision 2010-03-02</dc:source>. Субоператоры description в revision игнорируются.

## 10.50. Оператор rpc

Этот оператор отображается в показанную ниже ветвь схемы RELAX NG, где PREFIX - префикс локального модуля YANG.

```
<nma:rpc>
  <nma:input>
    <rng:element name="PREFIX:ARGUMENT">
      ... отображённое содержимое input ...
    </rng:element>
  </nma:input>
  <nma:output">
    ... отображённое содержимое output ...
  </nma:output>
</nma:rpc>
```

Как указано во фрагменте схемы, содержимое субоператора input (при наличии) отображается внутри <rng:element name="PREFIX: ARGUMENT">, а содержимое output - в <nma:output>. При отсутствии субоператора output включать элемент <nma:output> **недопустимо**. Элемент <nma:rpc> является потомком <nma:rpcs>.

## 10.51. Оператор status

Этот оператор может игнорироваться или отображаться в атрибут @nma:status со значением ARGUMENT.

## 10.52. Оператор submodule

Для этого оператора нет специального отображения, а его субоператоры отображаются, как будто они напрямую включены в модуль, к которому субмодуль относится.

## 10.53. Оператор type

Большинство встроенных типов YANG имеет эквиваленты в библиотеке типов данных XSD [XSD-D] (Таблица 4).

*Таблица 4. Встроенные типы данных YANG и их эквиваленты в W3C XML Schema Type Library.*

| *Тип YANG* | *Тип XSD* | *Назначение* |
|---|---|---|
| int8 | byte | 8-битовое целое число |
| int16 | short | 16-битовое целое число |
| int32 | int | 32-битовое целое число |
| int64 | long | 64-битовое целое число |
| uint8 | unsignedByte | 8-битовое целое число без знака |
| uint16 | unsignedShort | 16-битовое целое число без знака |
| uint32 | unsignedInt | 32-битовое целое число без знака |
| uint64 | unsignedLong | 64-битовое целое число без знака |
| string | string | Строка символов |
| binary | base64Binary | Двоичные данные в кодировке base64 |

Два важных типа из библиотеки типов данных XSD - dateTime и anyURI - не имеют встроенных аналогов в YANG и определены как производные типа в стандартных модулях [RFC6021]: date-and-time в модулей ietf-yang-types и uri - в ietf-inet-types. Однако формальные ограничения в определениях типов YANG достаточно слабы и реализациям отображений YANG-DSDL **следует** находить эти производные типы в исходных модулях YANG и отображать их в dateType и anyURI.

Отображения конкретных встроенных типов YANG описаны в последующих параграфах.

### *10.53.1. Тип empty*

Этот тип отображается в <rng:empty/>.

### *10.53.2. Тип boolean*

Этот встроенный тип не поддерживает ограничений и отображается в приведённый ниже фрагмент XML.

```
<rng:choice>
  <rng:value>true</rng:value>
  <rng:value>false</rng:value>
</rng:choice>
```

Отметим, что тип XSD boolean не может применяться здесь, поскольку он, в отличие от YANG, допускает численное представление логических значений - 0 для false и 1 для true.

### *10.53.3. Тип binary*

Этот встроенный тип не поддерживает ограничений и отображается вставкой элемента <rng:data> с атрибутом @type base64Binary (Таблица 4).

### *10.53.4. Тип bits*

Этот тип отображается в <rng:list> и для каждого субоператора bit вставляется показанный ниже фрагмент XML как потомок <rng:list>.

```
<rng:optional>
  <rng:value>bit_name</rng:value>
</rng:optional>
```

Здесь bit_name указывает имя бита из аргумента субоператора bit.

### *10.53.5. Типы enumeration и union*

Эти типы отображаются в элементы <rng:choice>.

### *10.53.6. Тип identityref*

Этот тип отображается в ссылку на именованный шаблон <rng:ref name="__PREFIX_BASE"/>, где PREFIX:BASE - полное имя идентификатора в аргументе субоператора base. Предположим, например, что модуль des из параграфа 10.21. Оператор identity содержит показанное ниже определение.

```
leaf foo {
  type identityref {
    base crypto:crypto-alg;
  }
}
```

Этот лист будет отображаться в шаблон

```
<element name="des:foo">
  <ref name="__crypto_crypto-alg"/>
</element>
```

### *10.53.7. Тип instance-identifier*

Этот тип отображается в элемент <rng:data> с атрибутом @type string. Кроме того, **должен** вставляться пустой элемент <nma:instance-identifier> как потомок PARENT. Аргументом субоператора require-instance (при наличии) будет значение атрибута @require-instance из элемента <nma:instance-identifier>.

### *10.53.8. Тип leafref*

Этот тип отображается так же как тип leaf в аргументе субоператора path. Однако если тип указанного листа задаёт принятое по умолчанию значение, оно должно игнорироваться при отображении. Кроме того, к PARENT **должен** добавляться атрибут @nma:leafref. Аргумент субоператора path, преобразованный в соответствии с параграфом 9.3. Трансляция выражений XPath, становится значением этого атрибута.

### *10.53.9. Численные типы*

Встроенными численными типами YANG являются int8, int16, int32, int64, uint8, uint16, uint32, uint64, decimal64. Они отображаются в элементы <rng:data> с атрибутом @type, имеющим значение ARGUMENT в соответствии с таблицей 4.

Тип decimal64 является исключением и отображается в тип decimal из библиотеки типов данных XSD. Его точность и число значащих цифр контролируются указанными ниже границами, которые **должны** присутствовать:

- totalDigits имеет значение 19;
- fractionDigits получает значение аргумента субоператора fraction-digits.

Фиксированное значение totalDigits соответствует максимальному числу десятичных цифр в 64-битовом целом числе.

Например оператор

```
type decimal64 {
    fraction-digits 2;
}
```

отображается в тип данных RELAX NG

```
<rng:data type="decimal">
  <rng:param name="totalDigits">19</rng:param>
  <rng:param name="fractionDigits">2</rng:param>
</rng:data>
```

Все численные типы поддерживают ограничение range, которое отображается, как указано ниже.

Если выражение содержит лишь один диапазон LO..HI, оно отображается в пару границ <rng:param name="minInclusive">LO</rng:param> и <rng:param name="maxInclusive">HI</rng:param>. Если диапазон указан одним числом, значения границ будут одинаковыми. Если LO имеет значение min, граница minInclusive опускается, а при HI со значением max опускается граница maxInclusive. Если диапазон содержит несколько блоков, разделённых |, элемент <rng:data> должен повторяться для каждого блока, а все такие элементы <rng:data> заключаются в <rng:choice>.

Каждый элемент <rng:data> содержит границы minInclusive и maxInclusive для одного блока, как указано выше. Для типа decimal64 значения totalDigits и fractionDigits должны повторяться в каждом элементе <rng:data>.

Например,

```
type int32 {
    range "-6378..0|42|100..max";
}
```

отображается во фрагмент RELAX NG

```
<rng:choice>
  <rng:data type="int">
    <rng:param name="minInclusive">-6378</rng:param>
    <rng:param name="maxInclusive">0</rng:param>
  </rng:data>
  <rng:data type="int">
    <rng:param name="minInclusive">42</rng:param>
    <rng:param name="maxInclusive">42</rng:param>
  </rng:data>
  <rng:data type="int">
    <rng:param name="minInclusive">100</rng:param>
  </rng:data>
</rng:choice>
```

Дополнительные детали и ограничения для отображения приведены в параграфе 9.2.2. Цепочки производных типов.

### *10.53.10. Тип string*

Этот тип отображается в элемент <rng:data> с атрибутом @type string. Ограничение length обрабатывается аналогично ограничению range для численных типов (10.53.9. Численные типы):

Если выражение для размер имеет просто один диапазон:

- при указании одного числа LENGTH используется <rng:param name="length">LENGTH</rng:param>;
- при указании в форме LO..HI (обе границы), вставляются элементы <rng:param name="minLength">LO</rng:param> и <rng:param name="maxLength">HI</rng:param>.

Если LO имеет значение min, граница minLength опускается, если HI имеет значение max, опускается maxLength. Если в выражении для размера имеется несколько частей, разделённых |, элемент <rng:data> должен применяться для каждой такой части, а все эти элементы <rng:data> заключаются в элемент <rng:choice>. Каждый элемент <rng:data> содержит значение length или границы minLength и maxLength для одной части ограничения.

Каждое ограничение pattern для одного типа данных string отображается в блок pattern

```
<rng:param name="pattern">...</rng:param>
```

с текстом, эквивалентным аргументу pattern. Такие блоки pattern должны повторяться внутри каждого элемента <rng:data>, т. е. по одному на каждую часть диапазона размера. Например,

```
type string {
    length "1|3..8";
    pattern "[A-Z][a-z]*";
}
```

отображается в приведённый ниже фрагмент RELAX NG.

```
<rng:choice>
  <rng:data type="string">
    <rng:param name="length">1</rng:param>
    <rng:param name="pattern">[A-Z][a-z]*</rng:param>
  </rng:data>
  <rng:data type="string">
    <rng:param name="minLength">3</rng:param>
    <rng:param name="maxLength">8</rng:param>
    <rng:param name="pattern">[A-Z][a-z]*</rng:param>
  </rng:data>
</rng:choice>
```

### *10.53.11. Производные типы*

Если оператор type указывает производный тип, он отображается одним из указанных ниже способов в зависимости от наличия ограничений в субоператорах.

1. Без ограничений type отображается в элемент <rng:ref> , т. е. ссылку на именованный шаблон. Если определения этого именованного шаблона ещё нет в гибридной схеме RELAX NG, **должно** быть найдено определение соответствующего типа и включено как субэлемент корневого или встроенного элемента <rng:grammar> (см. 10.54. Оператор typedef). Даже при неоднократном использовании производного типа во входных модулях YANG, соответствующее отображение typedef **должно** устанавливаться 1 раз.
2. При наличии ограничения нужно определить встроенного предка производного типа, а затем **должно** применяться отображение для этого базового типа. Все ограничения, указанные в цепочке производного типа **должны** учитываться, а их комбинация добавляется в элемент <rng:data>, определяющий базовый тип.

Дополнительные детали и пример приведены в параграфе 9.2. Отображение группировок и определений типов YANG.

## 10.54. Оператор typedef

Этот оператор отображается в определение именованного шаблона RELAX NG <rng:define>, но лишь при отсутствии ограничений при использовании заданного оператором типа во входных модулях. В этом случае определение именованного шаблона становится потомком корневого или встроенного элемента <rng:grammar> в зависимости от

присутствия оператора typedef на верхнем уровне модуля YANG. Именем этого определения шаблона будет значение ARGUMENT, преобразованное в соотсветствии с правилами параграфа 9.2. Отображение группировок и определений типов YANG.

При использовании производного типа с дополнительными ограничениями вместо него отображается встроенный родительский тип с ограничениями, являющимися сочетанием всех ограничений, заданных в цепочке создания типа. Пример и дополнительные детали приведены в параграфе 10.53.11. Производные типы.

Реализация **может** предложить запись всех операторов typedef как именованных шаблонов выходной схемы RELAX NG, даже если на эти типы не ссылаются. Это полезно для отображения библиотечных модулей YANG, содержащих лишь typedef и grouping.

## 10.55. Оператор unique

Этот оператор отображается в элемент <nma:unique>. Значение ARGUMENT **должно** транслироваться так, чтобы каждый идентификатор узла в каждом из его компонентов имел префикс пространства имён локального модуля, если такого префикса ещё нет. Результат трансляции будет значением аргумента @tag, присоединяемого к <nma:unique>[1].

Предположим, что локальный модуль имеет префикс ex. Тогда unique "foo ex:bar/baz" отображается в пару атрибут-значение nma:unique="ex:foo ex:bar/ex:baz"

## 10.56. Оператор units

Этот оператор отображается в атрибут @nma:units attribute and ARGUMENT becomes its value.

## 10.57. Оператор uses

При отсутствии субоператора refine или augment этот оператор отображается в элемент <rng:ref>, т. е. ссылку на именованный шаблон, а значением его атрибута @name будет ARGUMENT после преобразования в соответствии с параграфом 9.2. Отображение группировок и определений типов YANG. Если определения RELAX NG для этого именованного шаблона ещё нет, **должна** быть найдена соответствующая группировка с установкой её отображения как субэлемента <rng:grammar>, см. параграф 10.20. Оператор grouping. Если же оператор uses имеет субоператор refine или augment, должна отыскиваться соответствующая группировка со вставкой её содержимого в PARENT. Дополнительные детали и пример приведены в параграфе 9.2.1. Уточнения и дополнения YANG.

## 10.58. Оператор value

Этот оператор игнорируется.

## 10.59. Оператор when

Этот оператор отображается в атрибут @nma:when со значением ARGUMENT, преобразованным в соответствии с параграфом 9.3. Трансляция выражений XPath.

## 10.60. Оператор yang-version

Этот оператор не отображается в выходную схему. Однако реализации **следует** проверить совместимость с версией YANG, указанной в операторе (в настоящее время 1). При несоответствии реализации **следует** выдать сообщение об ошибке и прервать отображение.

## 10.61. Оператор yin-element

Этот оператор не отображается в выходную схему, но следует выполнять правила обработки расширений из параграфа 9.4. Расширения языка YANG.

# *11. Отображение гибридной схемы в DSDL*

Как указано в разделе 6, второй этап отображения YANG в DSDL принимает гибридную схему и преобразует её в различные схемы DSDL, пригодня для проверки экземпляров документов XML. В простейшем случае входным параметром этого этапа является просто спецификация типа проверяемого документа NETCONF XML. Таким типом может быть, например, содержимое хранилища данных, отклик на <nc:get> или <nc:get-config>, содержимое запросов и откликов RPC или уведомлений о событиях и т. п.

На этом этапе выполняются три основных задачи.

1. Извлечение частей гибридной схемы, соответствующих запрошенному типу документа, например, для отклика на <nc:get> это будет ветвь <nma:data>.
2. Адаптация схемы к конкретным элементам инкапсуляции XML, требуемым уровнем RPC. Например, это могут быть <nc:rpc> и <nc:data> для отклика на <nc:get> или <en:notification> для уведомления.
3. Сопоставление относящихся к NETMOD аннотаций, относящихся к отображаемому языку схемы с соответствующими шаблонами и правилами.

Эти 3 задачи вместе проще первого этапа отображения и могут быть реализованы с помощью преобразований XSL [XSLT]. В последующих параграфах подробно описан этот этап для отдельных языков схем DSDL. Затем в разделе 12 приведена подробная спецификация отображений всех аннотаций, специфичных для NETMOD.

## 11.1. Генерация схем RELAX NG для разных типов документов

За одним небольшим исключением, получение проверочной схемы RELAX NG из гибридной схемы состоит лишь в извлечении подходящих частей гибридной схемы и сборки из них новой грамматики RELAX NG (возможно, с удалением

[1]В оригинале этот абзац содержал ошибку, см. https://www.rfc-editor.org/errata/eid3362. *Прим. перев.*

ненужных аннотаций). Структура результирующей схемы RELAX NG похожа на структуру гибридной схемы - корневой блок (grammar) содержит встроенные блоки, по одному для каждого входного модуля YANG. Однако, как отмечено в параграфе 8.2. Модульность, определения глобальных именованных шаблонов (потомки корневого элемента <rng:grammar>) **должны** размещаться в отдельном файле схемы.

В зависимости от типа целевого документа XML для проверки, такого как отклик на <nc:get> или <nc:get-config>, операции RPC или уведомления, **должны** добавляться шаблоны, задающие соответствующие сведения верхнего уровня, такие как <nc:rpc-reply> с атрибутом @message-id и т. п.

Чтобы избежать копирования общих определений именованных шаблонов для базовых элементов и атрибутов NETCONF в каждый выходной файл RELAX NG, такие независимые от схемы определения **следует** собирать в файл библиотеки, включаемый затем в проверяемые схемы RELAX NG (пример такого файла представлен в Приложении B).

Незначительное исключение, отмеченное выше, - это аннотация @nma:config, которая должна присутствовать, если целевым типом документа является <nc:get-config>. В таком случае определения элементов, в которых этот атрибут имеет значение false, **должны** удаляться из схемы вместе с его потомками (см. 12.1. Аннотация @nma:config).

## 11.2. Отображение семантических ограничений в Schematron

Схемы Schematron обычно более плоские и однородные, нежели RELAX NG, и имеют ровно 4 уровня иерархии XML - <sch: schema>, <sch:pattern>, <sch:rule> и <sch:assert> или <sch:report>. В схемах Schematron, создаваемых на втором этапе базой организации является правило, представленное элементом <sch:rule>. В соответствующие правила Schematron из гибридной схемы отображаются связанные с NETMOD аннотации (далее, семантические аннотации) <nma:must>, <nma:unique>, @nma:key, @nma:max-elements, @nma:min-elements, @nma:when, @nma:leafref, @nma:leaf-list, а также @nma:mandatory как атрибут <rng: choice> (см. 11.2.1. Ограничения для обязательного choice)[1].

Каждый входной модуль YANG отображается в шаблон Schematron, атрибут @id которого получает имя модуля в качестве значения. Каждый шаблон <rng:element>, содержащий хотя бы 1 семантическую аннотацию, отображается в правило Schematron.

```
<sch:rule context="XELEM">
  ...
</sch:rule>
```

В качестве значения обязательного атрибута @context правила <sch:rule> (обозначается как XELEM) **должен** быть указан абсолютный путь к элементу контекста в дереве данных. Элемент <sch:rule> содержит отображения всех включённых семантических аннотаций в форме утверждений или отчетов Schematron.

Семантические аннотации в определении именованного шаблона (т. е. имеющие <rng:define> среди предков) требуют особого отношения, поскольку могут применяться в разном контексте. Это достигается за счёт применения разных абстрактных шаблонов Schematron, использующих переменную $pref вместо префикса локального пространства имён. Значением атрибута @id такого абстрактного шаблона **должно** быть имя определения шаблона, который отображается (т. е. преобразованное имя исходной группировки YANG). При создании экземпляра абстрактного шаблона **должны** предоставляться значения двух параметров:

- pref - фактический префикс пространства имён;
- start - выражение XPath, задающее контекст, где применяется группировка.

Рассмотрим в качестве примера приведённый ниже модуль YANG.

```
module example4 {
  namespace "http://example.com/ns/example4";
  prefix ex4;
  uses sorted-leaf-list;
  grouping sorted-leaf-list {
    leaf-list sorted-entry {
      must "not(preceding-sibling::sorted-entry > .)" {
        error-message "Записи должны быть упорядочены по возрастанию.";
      }
      type uint8;
    }
  }
}
```

Результирующая схема Schematron для отклика <nc:get> будет иметь вид

```
<?xml version="1.0" encoding="utf-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron">
  <sch:ns uri="http://example.com/ns/example4" prefix="ex4"/>
  <sch:ns uri="urn:ietf:params:xml:ns:netconf:base:1.0"
          prefix="nc"/>
  <sch:pattern abstract="true"
               id="_example4__sorted-leaf-list">
    <sch:rule context="$start/$pref:sorted-entry">
      <sch:report
          test=". = preceding-sibling::$pref:sorted-entry">
        Дубликат записи leaf-list "<sch:value-of select="."/>".
      </sch:report>
      <sch:assert
          test="not(preceding-sibling::$pref:sorted-entry &gt; .)">
        Записи должны быть упорядочены по возрастанию.
      </sch:assert>
    </sch:rule>
  </sch:pattern>
  <sch:pattern id="example4"/>
```

[1] В оригинале этот абзац содержал ошибку, см. https://www.rfc-editor.org/errata/eid3362. *Прим. перев.*

```
    <sch:pattern id="id2573371" is-a="_example4__sorted-leaf-list">
      <sch:param name="start" value="/nc:rpc-reply/nc:data"/>
      <sch:param name="pref" value="ex4"/>
    </sch:pattern>
  </sch:schema>
```

Группировка sorted-leaf-list из входного модуля отображается в абстрактный шаблон с атрибутом @id, имеющим значение _example4__sorted-leaf-list, в котором оператору must соответствует элемент <sch:assert>. Из абстрактного шаблона создаётся экземпляр с @id id2573371, который устанавливает значение параметров start и pref.

Отметим, что автоматически добавлен элемент Schematron <sch:report>, проверяющий дубликаты записей leaf-list.

Отображение гибридной схемы в Schematron выполняется за несколько этапов.

1. Выбирается активная ветвь (ветви) гибридной схемы в зависимости от запрошенного типа целевого документа. Эта процедура аналогична случаю RELAX NG, включая обработку @nma:config для целевого типа отклика на <nc:get-config>.
2. **Должны** объявляться пространства имён всех входных модулей YANG вместе с пространствами имён базового NETCONF (префикс nc ) или уведомлений (префикс en) с применением элемента <sch:ns>, например, <sch:ns uri="http://example.com/ns/example4" prefix="ex4"/>
3. Создаётся 1 шаблон для каждого входного модуля, а также абстрактные шаблоны, для каждого определения именованного шаблона, содержащего одну или несколько семантических аннотаций.
4. Создаётся элемент <sch:rule> для каждого шаблона элемента, содержащего семантические аннотации.
5. Каждая такая аннотация отображается в элемент <sch:assert> или <sch:report>, устанавливаемый как потомок элемента <sch:rule>.

### *11.2.1. Ограничения для обязательного choice*

Для полного представления семантики оператора YANG choice с субоператором mandatory true грамматика RELAX NG должна сочетаться со специальным правилом Schematron. Рассмотрим в качестве примера показанный ниже модуль.

```
module example5 {
    namespace "http://example.com/ns/example5";
    prefix ex5;
    choice foobar {
        mandatory true;
        case foo {
            leaf foo1 {
                type uint8;
            }
            leaf foo2 {
                type uint8;
            }
        }
        leaf bar {
            type uint8;
        }
    }
}
```

Здесь все три узла leaf в обеих ветвях case являются необязательными, но из-за оператора mandatory true в choice хотя бы один из этих узлов должен присутствовать в действительной конфигурации. Оператор choice из этого модуля отображается в приведённый ниже фрагмент схемы RELAX NG.

```
<rng:choice>
  <rng:interleave>
    <rng:optional>
      <rng:element name="ex5:foo1">
        <rng:data type="unsignedByte"/>
      </rng:element>
    </rng:optional>
    <rng:optional>
      <rng:element name="ex5:foo2">
        <rng:data type="unsignedByte"/>
      </rng:element>
    </rng:optional>
  </rng:interleave>
  <rng:element name="ex5:bar">
    <rng:data type="unsignedByte"/>
  </rng:element>
</rng:choice>
```

Во второй ветви case элемент ex5:bar определён как обязательный, поэтому он должен присутствовать в допустимой конфигурации, если эта ветвь выбрана. Однако оба элемента первой ветви foo не могут быть объявлены обязательными, поскольку любого из них достаточно для пригодной конфигурации. В результате приведённый выше фрагмент RELAX NG будет подтверждать конфигурации, где нет ни одного из трёх листьев.

Поэтому обязательные choice, которые могут быть распознаны в гибридной схеме как элементы <rng:choice> с аннотацией @nma:mandatory, обрабатываются особым способом. Для каждого обязательного choice, где хотя бы один из вариантов содержит более 1 узла, **должно** добавляться правило Schematron, обеспечивающее наличие хотя бы одного узла из любого такого case (схема RELAX NG гарантирует, что элементы из разных case не перемешиваются, обязательные узлы присутствуют и т. д.).

Для приведённого выше примера правило Schematron будет иметь вид

```
<sch:rule context="/nc:rpc-reply/nc:data">
```

```
    <sch:assert test="ex5:foo1 or ex5:foo2 or ex5:bar">
      Должен быть узел хотя бы из 1 case оператора choice foobar.
    </sch:assert>
  </sch:rule>
```

## 11.3. Отображение принятых по умолчанию значений в DSRL

DSRL - единственный компонент DSDL, которому разрешено менять набор информации в проверенных документах XML. Хотя в DSRL имеются и другие функции, отображение YANG-DSDL использует лишь задание и применение принятого по умолчанию содержимого. Для экземпляров документов XML, основанных на моделях данных YANG, вставка принятого по умолчанию содержимого может выполняться для всех неявных узлов, идентифицированных по правилам параграфа 9.2.1. Уточнения и дополнения YANG.

В DSRL принятое по умолчанию содержимое элемента задаётся с использованием элемента <dsrl:default-content>, являющегося потомком <dsrl:element-map>. Два братских (sibling) элемента <dsrl:default-content> определяют контекст для применения заданного по умолчанию содержимого (см. [DSRL]):

- элемент <dsrl:parent> содержит шаблон XSLT, задающий родительский элемент и заданное по умолчанию содержимое применяется лишь при наличии этого родителя в экземпляре документа;
- <dsrl:name> содержит имя XML элемента, который, будучи отсутствующим или пустым, вставляется вместе с содержимым <dsrl: default-content>.

Элемент <dsrl:parent> является необязательным в базовой схеме DSRL, но для отображения YANG-DSDL этот элемент **должен** присутствовать для гарантии корректного применения заданного по умолчанию содержимого.

Отображение DSRL работает лишь с определяющими неявные узлы шаблонами <rng:element> из гибридной схемы (см. 9.1.2. Неявные узлы). Такие шаблоны элементов отличаются наличием специфичных для NETMOD атрибутов аннотаций @nma:default или @nma:implicit, т. е.

```
  <rng:element name="ELEM" nma:default="DEFVALUE">
    ...
  </rng:element>
```

или

```
  <rng:element name="ELEM" nma:implicit="true">
    ...
  </rng:element>
```

Первый вариант применяется к узлам leaf, имеющим субоператор default, а также к узлам leaf, получающим заданные по умолчанию значения от typedef, если это определение типа преобразуется по правилам параграфа 9.2. Отображение группировок и определений типов YANG, так что аннотация @nma:default присоединяется напрямую к шаблону элемента. Второй вариант применяется для всех неявных контейнеров (9.1. Правила вхождения для узлов данных) и leaf, которые получают заданное по умолчанию значение от typedef и не имеют аннотации @nma:default.

В простейшем случае оба шаблона элементов отображаются в приведённый ниже фрагмент DSRL.

```
  <dsrl:element-map>
    <dsrl:parent>XPARENT</dsrl:parent>
    <dsrl:name>ELEM</dsrl:name>
    <dsrl:default-content>DEFCONT</dsrl:default-content>
  </dsrl:element-map>
```

где XPARENT — абсолютный XPath родительского элемента ELEM в дереве данных, а DEFCONT строится, как показано ниже.

- Если неявный узел ELEM является листом (leaf) и имеет атрибут @nma:default, для DEFCONT устанавливается значение этого атрибута (DEFVALUE).
- Если неявный узел ELEM является листом и имеет атрибут @nma:implicit со значением true, принятое по умолчанию значение определяется из атрибута @nma:default в определении типа ELEM (возможно, рекурсивно) и применяется вместо DEFCONT в приведённом выше элементе DSRL (см. 9.2.2. Цепочки производных типов).
- В остальных случаях неявный узел ELEM является контейнером и DEFCONT строится как фрагмент XML, содержащий все элементы-потомки ELEM с атрибутом @nma:implicit или @nma:default.

Кроме того, при отображении принятого по умолчанию case из оператора choice нужно гаратировать, что принятое по умолчанию содержимое не применяется, если имеется какой-либо из вариантов (case), не заданных по умолчанию. Это достигается установкой <dsrl:parent> особым способом <dsrl:parent>XPARENT[not (ELEM1|ELEM2|...|ELEMn)]</dsrl:parent>, где ELEM1, ELEM2, ... ELEMn - имена всех узлов верхнего уровня из всех case, на заданных по умолчанию. Остальная часть элемента не меняется.

Рассмотрим в качестве примера модуль YANG, показанный ниже.

```
module example6 {
  namespace "http://example.com/ns/example6";
  prefix ex6;
  container outer {
    leaf leaf1 {
      type uint8;
      default 1;
    }
    choice one-or-two {
      default "one";
      container one {
        leaf leaf2 {
          type uint8;
          default 2;
```

```
          }
        }
        leaf leaf3 {
          type uint8;
          default 3;
        }
      }
    }
  }
```

Схема DSRL для целевого документа get-reply имеет вид

```
<?xml version="1.0" encoding="utf-8"?>
<dsrl:maps xmlns:dsrl="http://purl.oclc.org/dsdl/dsrl"
           xmlns:ex6="http://example.com/ns/example6"
           xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <dsrl:element-map>
    <dsrl:parent>/nc:rpc-reply/nc:data</dsrl:parent>
    <dsrl:name>ex6:outer</dsrl:name>
    <dsrl:default-content>
      <ex6:leaf1>1</ex6:leaf1>
      <ex6:one>
        <ex6:leaf2>2</ex6:leaf2>
      </ex6:one>
    </dsrl:default-content>
  </dsrl:element-map>
  <dsrl:element-map>
    <dsrl:parent>/nc:rpc-reply/nc:data/ex6:outer</dsrl:parent>
    <dsrl:name>ex6:leaf1</dsrl:name>
    <dsrl:default-content>1</dsrl:default-content>
  </dsrl:element-map>
  <dsrl:element-map>
    <dsrl:parent>
      /nc:rpc-reply/nc:data/ex6:outer[not(ex6:leaf3)]
    </dsrl:parent>
    <dsrl:name>ex6:one</dsrl:name>
    <dsrl:default-content>
      <ex6:leaf2>2</ex6:leaf2>
    </dsrl:default-content>
  </dsrl:element-map>
  <dsrl:element-map>
    <dsrl:parent>
      /nc:rpc-reply/nc:data/ex6:outer/ex6:one
    </dsrl:parent>
    <dsrl:name>ex6:leaf2</dsrl:name>
    <dsrl:default-content>2</dsrl:default-content>
  </dsrl:element-map>
</dsrl:maps>
```

Отметим, что принятое по умолчанию значение для leaf3, заданное в модуле YANG, игнорируется, поскольку leaf3 не представляет принятый по умолчанию вариант choice и поэтому не становится неявным элементом.

# *12. Отображение аннотаций NETMOD в языки схем DSDL*

В этом разделе приведены спецификации отображений для отдельных аннотаций, специфичных для NETMOD. В каждом случае результат отображения должен помещаться в подходящий контекст целевой схемы DSDL, как описано в разделе 11. Отображение гибридной схемы в DSDL. Контекст определяется шаблоном элемента в гибридной схеме, к которому присоединена аннотация. В этом разделе CONTELEM указывает имя этого элемента контекста с префиксом пространства имён.

## 12.1. Аннотация @nma:config

Если эта аннотация присутствует со значением false, **должны** соблюдаться приведённые ниже правила для схем DSDL откликов на <nc:get-config>.

- При генерации RELAX NG содержимое определения CONTELEM **должно** заменяться на <rng:notAllowed>.
- При генерации Schematron или DSRL определение CONTELEM и все его потомки в гибридной схеме **должны** игнорироваться.

## 12.2. Аннотация @nma:default

Эта аннотация применяется для генерации схемы DSRL, как описано в параграфе 11.3. Отображение принятых по умолчанию значений в DSRL.

## 12.3. Аннотация <nma:error-app-tag>

Для этой аннотации с настоящее время не задано отображения.

## 12.4. Аннотация <nma:error-message>

Эта аннотация обрабатывается в рамках <nma:must>, см. параграф 12.13. Аннотация <nma:must>.

## 12.5. Аннотация @if-feature

Сведения о доступных свойствах (feature) **могут** предоставляться в качестве входного параметра для реализации. В этом случае **должны** вноситься указанные ниже изменения для всех свойств, сочтённых недоступными.

- При генерации RELAX NG CONTELEM **должно** заменяться на <rng:notAllowed>.
- При генерации Schematron или DSRL определение CONTELEM и все его потомки в гибридной схеме должны игнорироваться.

## 12.6. Аннотация @nma:implicit

Эта аннотация применяется для генерации схемы DSRL, как описано в параграфе 11.3. Отображение принятых по умолчанию значений в DSRL.

## 12.7. Аннотация <nma:instance-identifier>

Если этот элемент аннотации имеет атрибут @require-instance со значением false, он игнорируется. В ином случае элемент отображается в утверждение Schematron

```
<sch:assert test="nmf:evaluate(.)">
  Элемент, указанный CONTELEM, должен существовать.
</sch:assert>
```

Функция nmf:evaluate() является функцией преобразования XSLT (см. Extension Functions в [XSLT]), которая оценивает выражение XPath во время исполнения. Такая функция расширения доступна в Extended XSLT (EXSLT) или предоставляется как фирменное (proprietary) расширение некоторыми процессорами XSLT (например, Saxon).

## 12.8. Аннотация @nma:key

Предположим, что этот атрибут аннотации содержит k_1 k_2 ... k_n, т. е. задаёт n потомков CONTELEM как ключи списка. Тогда аннотация отображается в показанный ниже отчёт Schematron.

```
<sch:report test="CONDITION">
  Дубликат ключа списка CONTELEM
</sch:report>
```

где CONDITION имеет вид preceding-sibling::CONTELEM[C_1 and C_2 and ... and C_n]. Каждое субвыражение C_i для i=1,2,...,n задаёт условие неуникальности ключа k_i, а именно k_i=current()/k_i.

## 12.9. Аннотация @nma:leaf-list

Эта аннотация отображается в приведённое ниже правило Schematron, которое находит дубликаты записей в leaf-list.

```
<sch:report
    test=". = preceding-sibling::PREFIX:sorted-entry">
  Дубликат записи leaf-list "<sch:value-of select="."/>".
</sch:report>
```

Полный пример представлен в параграфе 11.2. Отображение семантических ограничений в Schematron.

## 12.10. Аннотация @nma:leafref

Эта аннотация отображается в приведённое ниже утверждение.

```
<sch:assert test="PATH=.">
  Листа PATH не существует для leafref со значением VALUE
</sch:assert>
```

где PATH - значение @nma:leafref, VALUE - значение элемента контекста в экземпляре документа, для которого указанный лист (leaf) не существует.

## 12.11. Аннотация @nma:min-elements

Эта аннотация отображается в приведённое ниже утверждение Schematron.

```
<sch:assert test="count(../CONTELEM)&gt;=MIN">
  Список CONTELEM — число элементов должно быть не меньше MIN
</sch:assert>
```

где MIN - значение @nma:min-elements.

## 12.12. Аннотация @nma:max-elements

Эта аннотация отображается в приведённое ниже утверждение Schematron.

```
<sch:assert
    test="count(../CONTELEM)&lt;=MAX or preceding-sibling::../CONTELEM">
  Число элементов списка должно быть не больше MAX
</sch:assert>
```

где MAX - значение @nma:max-elements.

## 12.13. Аннотация <nma:must>

Эта аннотация отображается в приведённое ниже Schematron.

```
<sch:assert test="EXPRESSION">
  MESSAGE
</sch:assert>
```

где EXPRESSION - значение атрибута @assert для <nma:must>. Если субэлемент <nma:error-message> существует, в MESSAGE помещается его содержимое, иначе - принятое по умолчанию сообщение Condition EXPRESSION must be true.

## 12.14. Аннотация <nma:ordered-by>

Для этой аннотации с настоящее время не задано отображения.

## 12.15. Аннотация <nma:status>

Для этой аннотации с настоящее время не задано отображения.

## 12.16. Аннотация <nma:unique>[1]

Отображение этой аннотации похоже на @nma:key (12.7. Аннотация <nma:instance-identifier>) с двумя отличиями.

- Значением атрибута @tag для <nma:unique> является список идентификаторов узлов-потомков, а не имён простых leaf. Обнако выражение XPath, заданное в параграфе 12.8, работает без изменений, если идентификаторы узлов-потомков заменяются на k_1, k_2, ..., k_n.
- Сообщение, отображаемое как текст <sch:report>, отличается - Violated uniqueness of NODES (нарушена уникальность NODES), где NODES - значение атрибута @tag.

## 12.17. Аннотация @nma:when

Эта аннотация отображается в приведённое ниже утверждение Schematron.

```
<sch:assert test="EXPRESSION">
  Узел CONTELEM действителен лишь при EXPRESSION true.
</sch:assert>
```

где EXPRESSION - значение @nma:when.

# *13. Взаимодействие с IANA*

Этот документ запрашивает регистрацию двух URI пространств имён в реестре IETF XML [RFC3688]

```
URI: urn:ietf:params:xml:ns:netmod:dsdl-annotations:1
Registrant Contact: The IESG.
XML: N/A, запрошенный URI является пространством имён XML.

URI: urn:ietf:params:xml:ns:netmod:xpath-extensions:1
Registrant Contact: The IESG.
XML: N/A, запрошенный URI является пространством имён XML.
```

# *14. Вопросы безопасности*

Этот документ определяет процедуру отображения моделей данных на языке YANG в согласованный набор схем DSDL. Процедура сама по себе не влияет на безопасность Internet.

Схемы DSDL, получаемые при отображении, могут применяться для проверки содержимого сообщений NETCONF или целых хранилищ данных, что позволяет дополнительно проверить достоверность в дополнение к проверка в реализациях клиентов и серверов NETCONF, поддерживающих модели данных YANG. Строгость такой проверки напрямую зависит от исходных модулей YANG, которым соответствуют проверенные данные XML.

# *15. Участники работы*

Ниже указаны люди, внёсшие значительный вклад в начальный вариант этого документа.

- Rohan Mahy
- Sharon Chisholm (Ciena)

# *16. Благодарности*

Редактор хотел бы поблагодарить за предоставленные полезные предложения и комментарии к разным версиям этого документа Andy Bierman, Martin Bjorklund, Jirka Kosek, Juergen Schoenwaelder, Phil Shafer.

# *17. Литература*

## 17.1. Нормативные документы

[DSDL] ISO/IEC, "Document Schema Definition Languages (DSDL) - Part 1: Overview", ISO/IEC 19757-1, November 2004.

[DSRL] ISO/IEC, "Information Technology - Document Schema Definition Languages (DSDL) - Part 8: Document Semantics Renaming Language - DSRL", ISO/IEC 19757-8:2008(E), December 2008.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC4741] Enns, R., "NETCONF Configuration Protocol", RFC 4741, December 2006.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6021] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6021, October 2010.

[RNG] ISO/IEC, "Information Technology - Document Schema Definition Languages (DSDL) - Part 2: Regular-Grammar-Based Validation - RELAX NG. Second Edition.", ISO/IEC 19757-2:2008(E), December 2008.

[RNG-CS] ISO/IEC, "Information Technology - Document Schema Definition Languages (DSDL) - Part 2: Regular-Grammar-Based Validation - RELAX NG. AMENDMENT 1: Compact Syntax", ISO/IEC 19757-2:2003/Amd. 1:2006(E), January 2006.

[1] В оригинале этот параграф содержал ошибки, см. https://www.rfc-editor.org/errata/eid3362. *Прим. перев.*

[RNG-DTD] Clark, J., Ed. and M. Murata, Ed., "RELAX NG DTD Compatibility", OASIS Committee Specification, 3 December 2001.

[Schematron] ISO/IEC, "Information Technology - Document Schema Definition Languages (DSDL) - Part 3: Rule-Based Validation - Schematron", ISO/IEC 19757-3:2006(E), June 2006.

[XML] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <http://www.w3.org/TR/2006/REC-xml-20060816>.

[XML-INFOSET] Tobin, R. and J. Cowan, "XML Information Set (Second Edition)", World Wide Web Consortium Recommendation REC-xml-infoset-20040204, February 2004, <http://www.w3.org/TR/2004/REC-xml-infoset-20040204>.

[XPath] Clark, J. and S. DeRose, "XML Path Language (XPath) Version 1.0", World Wide Web Consortium Recommendation REC-xpath-19991116, November 1999, <http://www.w3.org/TR/1999/REC-xpath-19991116>.

[XSD-D] Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

[XSLT] Clark, J., "XSL Transformations (XSLT) Version 1.0", World Wide Web Consortium Recommendation REC-xslt-19991116, November 1999.

## 17.2. Дополнительная литература

[DHCPtut] Bjorklund, M., "DHCP Tutorial", November 2007, <http://www.yang-central.org/twiki/bin/view/Main/DhcpTutorial>.

[RFC1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1157, May 1990.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC5013] Kunze, J., "The Dublin Core Metadata Element Set", RFC 5013, August 2007.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, July 2008.

[Trang] Clark, J., "Trang: Multiformat schema converter based on RELAX NG", 2008, <http://www.thaiopensource.com/relaxng/trang.html>.

[Vli04] van der Vlist, E., "RELAX NG", O'Reilly, 2004.

[XSD] Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>.

[pyang] Bjorklund, M. and L. Lhotka, "pyang: An extensible YANG validator and converter in Python", 2010, <http://code.google.com/p/pyang/>.

# *Приложение A. Схема RELAX NG для связанных с NETMOD аннотаций*

В этом приложении задана модель содержимого для всех относящихся к NETMOD аннотаций в форме определений именованных шаблонов RELAX NG.

```
<CODE BEGINS> file "nmannot.rng"

<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:nma="urn:ietf:params:xml:ns:netmod:dsdl-annotations:1"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

<define name="config-attribute">
  <attribute name="nma:config">
    <data type="boolean"/>
  </attribute>
</define>

<define name="data-element">
  <element name="nma:data">
    <ref name="__anyxml__"/>
  </element>
</define>

<define name="default-attribute">
  <attribute name="nma:default">
    <data type="string"/>
  </attribute>
</define>

<define name="error-app-tag-element">
  <element name="nma:error-app-tag">
    <text/>
  </element>
</define>
```

```
  <define name="error-message-element">
    <element name="nma:error-message">
      <text/>
    </element>
  </define>

  <define name="if-feature-attribute">
    <attribute name="nma:if-feature">
      <list>
        <data type="QName"/>
      </list>
    </attribute>
  </define>

  <define name="implicit-attribute">
    <attribute name="nma:implicit">
      <data type="boolean"/>
    </attribute>
  </define>

  <define name="instance-identifier-element">
    <element name="nma:instance-identifier">
      <optional>
        <attribute name="nma:require-instance">
          <data type="boolean"/>
        </attribute>
      </optional>
    </element>
  </define>

  <define name="key-attribute">
    <attribute name="nma:key">
      <list>
        <data type="QName"/>
      </list>
    </attribute>
  </define>

  <define name="leaf-list-attribute">
    <attribute name="nma:leaf-list">
      <data type="boolean"/>
    </attribute>
  </define>

  <define name="leafref-attribute">
    <attribute name="nma:leafref">
      <data type="string"/>
    </attribute>
  </define>

  <define name="mandatory-attribute">
    <attribute name="nma:mandatory">
      <data type="Name"/>
    </attribute>
  </define>

  <define name="max-elements-attribute">
    <attribute name="nma:max-elements">
      <data type="nonNegativeInteger"/>
    </attribute>
  </define>

  <define name="min-elements-attribute">
    <attribute name="nma:min-elements">
      <data type="nonNegativeInteger"/>
    </attribute>
  </define>

  <define name="module-attribute">
    <attribute name="nma:module">
      <data type="Name"/>
    </attribute>
  </define>

  <define name="must-element">
    <element name="nma:must">
      <attribute name="assert">
        <data type="string"/>
      </attribute>
      <interleave>
        <optional>
          <ref name="error-app-tag-element"/>
        </optional>
        <optional>
          <ref name="error-message-element"/>
        </optional>
```

```
      </interleave>
    </element>
  </define>

  <define name="notifications-element">
    <element name="nma:notifications">
      <zeroOrMore>
        <element name="nma:notification">
          <ref name="__anyxml__"/>
        </element>
      </zeroOrMore>
    </element>
  </define>

  <define name="rpcs-element">
    <element name="nma:rpcs">
      <zeroOrMore>
        <element name="nma:rpc">
          <interleave>
            <element name="nma:input">
              <ref name="__anyxml__"/>
            </element>
            <optional>
              <element name="nma:output">
                <ref name="__anyxml__"/>
              </element>
            </optional>
          </interleave>
        </element>
      </zeroOrMore>
    </element>
  </define>

  <define name="ordered-by-attribute">
    <attribute name="nma:ordered-by">
      <choice>
        <value>user</value>
        <value>system</value>
      </choice>
    </attribute>
  </define>

  <define name="status-attribute">
    <optional>
      <attribute name="nma:status">
        <choice>
          <value>current</value>
          <value>deprecated</value>
          <value>obsolete</value>
        </choice>
      </attribute>
    </optional>
  </define>

  <define name="unique-element">
    <element name="nma:unique">
      <attribute name="tag">
        <list>
          <data type="token"/>
        </list>
      </attribute>
    </element>
  </define>
  <define name="units-attribute">[1]
    <optional>
      <attribute name="nma:units">
        <data type="string"/>
      </attribute>
    </optional>
  </define>

  <define name="when-attribute">
    <optional>
      <attribute name="nma:when">
        <data type="string"/>
      </attribute>
    </optional>
  </define>

  <define name="__anyxml__">
    <zeroOrMore>
      <choice>
        <attribute>
          <anyName/>
```

[1] В оригинале этот блок содержал ошибку, см. https://www.rfc-editor.org/errata/eid3362. *Прим. перев.*

```
      </attribute>
      <element>
        <anyName/>
        <ref name="__anyxml__"/>
      </element>
      <text/>
    </choice>
  </zeroOrMore>
</define>

</grammar>

<CODE ENDS>
```

## *Приложение B. Независимая от схем библиотека*

Во избежание копирования базовых определений именованных шаблонов в каждую создаваемую на втором этапе схему RELAX NG определения собраны в файлы независимой от схемы библиотеки, которые включаются проверяющими схемами по имени relaxng-lib.rng (синтаксис XML) и relaxng-lib.rnc (компактный синтаксис). Включённые определения охватывают шаблоны для основных элементов из базового NETCONF [RFC4741] и уведомления о событиях [RFC5277].

```
<CODE BEGINS> file "relaxng-lib.rng"

<?xml version="1.0" encoding="UTF-8"?>

<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
         xmlns:en="urn:ietf:params:xml:ns:netconf:notification:1.0"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

  <define name="message-id-attribute">
    <attribute name="message-id">
      <data type="string">
        <param name="maxLength">4095</param>
      </data>
    </attribute>
  </define>

  <define name="ok-element">
    <element name="nc:ok">
      <empty/>
    </element>
  </define>

  <define name="eventTime-element">
    <element name="en:eventTime">
      <data type="dateTime"/>
    </element>
  </define>
</grammar>

<CODE ENDS>
```

## *Приложение C. Сопоставление модели данных DHCP - полный пример*

В этом приложении показаны оба этапа отображения YANG-DSDL для «канонической» модели данных из учебника по DHCP [DHCPtut]. Входной модуль YANG представлен в Приложении C.1, а выходные схемы - в двух последующих.

Гибридная схема была получена с помощью плагина dsdl для pyang [pyang], а проверка схем DSDL была выполнена с помощью таблиц стилей XSLT, также входящих в пакет pyang.

Из-за ограничения размера строк (72 символа) некоторые длинные строки были отредактированы вручную (шаблоны регулярных выражений для адресов IP и т. п.). В некоторые строки были добавлены символы переноса \ (документация и сообщения Schematron). В остальном результаты автоматической трансляции не изменялись.

## C.1. Входной модуль YANG

```
module dhcp {
  namespace "http://example.com/ns/dhcp";
  prefix dhcp;

  import ietf-yang-types { prefix yang; }
  import ietf-inet-types { prefix inet; }

  organization
    "yang-central.org";
  description
    "Частичная модель данных DHCP на основе эталонной реализации
     ISC DHCP.";

  container dhcp {
    description
      "Конфигурационные и рабочие параметры сервера DHCP.";

    leaf max-lease-time {
      type uint32;
```

```
        units seconds;
        default 7200;
      }

      leaf default-lease-time {
        type uint32;
        units seconds;
        must '. <= ../max-lease-time' {
          error-message
            "default-lease-time должно быть не больше max-lease-time";
        }
        default 600;
      }

      uses subnet-list;

      container shared-networks {
        list shared-network {
          key name;

          leaf name {
            type string;
          }
          uses subnet-list;
        }
      }

      container status {
        config false;
        list leases {
          key address;
          leaf address {
            type inet:ip-address;
          }
          leaf starts {
            type yang:date-and-time;
          }
          leaf ends {
            type yang:date-and-time;
          }
          container hardware {
            leaf type {
              type enumeration {
                enum "ethernet";
                enum "token-ring";
                enum "fddi";
              }
            }
            leaf address {
              type yang:phys-address;
            }
          }
        }
      }
    }

    grouping subnet-list {
      description "Многократно применяемый список подсетей";
      list subnet {
        key net;
        leaf net {
          type inet:ip-prefix;
        }
        container range {
          presence "Разрешает динамическое назначение адресов";
          leaf dynamic-bootp {
            type empty;
            description
              "Разрешает клиентам BOOTP получать адреса из блока";
          }
          leaf low {
            type inet:ip-address;
            mandatory true;
          }
          leaf high {
            type inet:ip-address;
            mandatory true;
          }
        }
        container dhcp-options {
          description "Опции протокола DHCP";
          leaf-list router {
            type inet:host;
            ordered-by user;
            reference "Параграф 3.8 в RFC 2132";
          }
```

```
          leaf domain-name {
            type inet:domain-name;
            reference "Параграф 3.17 в RFC 2132";
          }
        }

        leaf max-lease-time {
          type uint32;
          units seconds;
          default 7200;
        }
      }
    }
  }
```

## C.2. Гибридная схема

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar
    xmlns="http://relaxng.org/ns/structure/1.0"
    xmlns:nma="urn:ietf:params:xml:ns:netmod:dsdl-annotations:1"
    xmlns:dc="http://purl.org/dc/terms"
    xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
    xmlns:dhcp="http://example.com/ns/dhcp"
    datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
 <dc:creator>Pyang 1.0a, DSDL plugin</dc:creator>
 <dc:date>2010-06-17</dc:date>
 <start>
  <grammar nma:module="dhcp" ns="http://example.com/ns/dhcp">
   <dc:source>YANG module 'dhcp'</dc:source>
   <start>
    <nma:data>
     <optional>
      <element nma:implicit="true" name="dhcp:dhcp">
       <interleave>
        <a:documentation>
         Конфигурационные и рабочие параметры сервера DHCP.
        </a:documentation>
        <optional>
         <element nma:default="7200"
                  name="dhcp:max-lease-time"
                  nma:units="seconds">
          <data type="unsignedInt"/>
         </element>
        </optional>
        <optional>
         <element nma:default="600"
                  name="dhcp:default-lease-time"
                  nma:units="seconds">
          <data type="unsignedInt"/>
          <nma:must assert=". &lt;= ../dhcp:max-lease-time">
           <nma:error-message>
            default-lease-time должно быть не больше max-lease-time
           </nma:error-message>
          </nma:must>
         </element>
        </optional>
        <ref name="_dhcp__subnet-list"/>
        <optional>
         <element name="dhcp:shared-networks">
          <zeroOrMore>
           <element nma:key="dhcp:name"
                    name="dhcp:shared-network">
            <element name="dhcp:name">
             <data type="string"/>
            </element>
            <ref name="_dhcp__subnet-list"/>
           </element>
          </zeroOrMore>
         </element>
        </optional>
        <optional>
         <element name="dhcp:status" nma:config="false">
          <zeroOrMore>
           <element nma:key="dhcp:address"
                    name="dhcp:leases">
            <element name="dhcp:address">
             <ref name="ietf-inet-types__ip-address"/>
            </element>
            <interleave>
             <optional>
              <element name="dhcp:starts">
               <ref name="ietf-yang-types__date-and-time"/>
              </element>
             </optional>
             <optional>
```

```
                <element name="dhcp:ends">
                 <ref name="ietf-yang-types__date-and-time"/>
                </element>
               </optional>
               <optional>
                <element name="dhcp:hardware">
                 <interleave>
                  <optional>
                   <element name="dhcp:type">
                    <choice>
                     <value>ethernet</value>
                     <value>token-ring</value>
                     <value>fddi</value>
                    </choice>
                   </element>
                  </optional>
                  <optional>
                   <element name="dhcp:address">
                    <ref name="ietf-yang-types__phys-address"/>
                   </element>
                  </optional>
                 </interleave>
                </element>
               </optional>
              </interleave>
             </element>
            </zeroOrMore>
           </element>
          </optional>
         </interleave>
        </element>
       </optional>
     </nma:data>
     <nma:rpcs/>
     <nma:notifications/>
    </start>
   </grammar>
  </start>
  <define name="ietf-yang-types__phys-address">
   <data type="string">
    <param name="pattern">([0-9a-fA-F]{2}(:[0-9a-fA-F]{2})*)?</param>
   </data>
  </define>
  <define name="ietf-inet-types__ipv6-address">
   <data type="string">
    <param name="pattern">... шаблон regex ...</param>
    <param name="pattern">... шаблон regex ...</param>
   </data>
  </define>
  <define name="ietf-inet-types__ip-prefix">
   <choice>
    <ref name="ietf-inet-types__ipv4-prefix"/>
    <ref name="ietf-inet-types__ipv6-prefix"/>
   </choice>
  </define>
  <define name="ietf-inet-types__host">
   <choice>
    <ref name="ietf-inet-types__ip-address"/>
    <ref name="ietf-inet-types__domain-name"/>
   </choice>
  </define>
  <define name="ietf-yang-types__date-and-time">
   <data type="string">
    <param name="pattern">... шаблон regex ...</param>
   </data>
  </define>
  <define name="_dhcp__subnet-list">
   <a:documentation>Многократно применяемый список подсетей
   </a:documentation>
   <zeroOrMore>
    <element nma:key="net" name="subnet">
     <element name="net">
      <ref name="ietf-inet-types__ip-prefix"/>
     </element>
     <interleave>
      <optional>
       <element name="range">
        <interleave>
         <optional>
          <element name="dynamic-bootp">
           <a:documentation>
            Разрешает клиентам BOOTP получать адреса из блока
           </a:documentation>
           <empty/>
          </element>
         </optional>
```

```
          <element name="low">
           <ref name="ietf-inet-types__ip-address"/>
          </element>
          <element name="high">
           <ref name="ietf-inet-types__ip-address"/>
          </element>
         </interleave>
        </element>
       </optional>
       <optional>
        <element name="dhcp-options">
         <interleave>
          <a:documentation>
           Опции протокола DHCP
          </a:documentation>
          <zeroOrMore>
           <element nma:leaf-list="true" name="router"
                    nma:ordered-by="user">
            <a:documentation>
             См. параграф 3.8 в RFC 2132
            </a:documentation>
            <ref name="ietf-inet-types__host"/>
           </element>
          </zeroOrMore>
          <optional>
           <element name="domain-name">
            <a:documentation>
              См. параграф 3.17 в RFC 2132
            </a:documentation>
            <ref name="ietf-inet-types__domain-name"/>
           </element>
          </optional>
         </interleave>
        </element>
       </optional>
       <optional>
        <element nma:default="7200" name="max-lease-time"
                 nma:units="seconds">
         <data type="unsignedInt"/>
        </element>
       </optional>
      </interleave>
     </element>
    </zeroOrMore>
   </define>
   <define name="ietf-inet-types__domain-name">
    <data type="string">
     <param name="pattern">... шаблон regex ...</param>
     <param name="minLength">1</param>
     <param name="maxLength">253</param>
    </data>
   </define>
   <define name="ietf-inet-types__ipv4-prefix">
    <data type="string">
     <param name="pattern">... шаблон regex ...</param>
    </data>
   </define>
   <define name="ietf-inet-types__ipv4-address">
    <data type="string">
     <param name="pattern">... шаблон regex ...</param>
    </data>
   </define>
   <define name="ietf-inet-types__ipv6-prefix">
    <data type="string">
     <param name="pattern">... шаблон regex ...</param>
     <param name="pattern">... шаблон regex ...</param>
    </data>
   </define>
   <define name="ietf-inet-types__ip-address">
    <choice>
     <ref name="ietf-inet-types__ipv4-address"/>
     <ref name="ietf-inet-types__ipv6-address"/>
    </choice>
   </define>
  </grammar>
```

## C.3. Окончательные схемы DSDL

Здесь представлены схемы DSDL, полученные из гибридной схемы в Приложении C.2 путём преобразования XSL. Эти схемы можно применять напрямую для проверки отклика на нефильтрованный запрос <nc:get> с содержимым, соответствующим модели данных DHCP. Схема RELAX NG (из 2 частей, как указано в параграфе 8.2. Модульность) включает также независимую от схемы библиотеку из Приложения B.

### *C.3.1. Основная схема RELAX NG для отклика <nc:get>*

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<grammar
    xmlns="http://relaxng.org/ns/structure/1.0"
    xmlns:nma="urn:ietf:params:xml:ns:netmod:dsdl-annotations:1"
    xmlns:dhcp="http://example.com/ns/dhcp"
    datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"
    ns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <include href="relaxng-lib.rng"/>
 <start>
  <element name="rpc-reply">
   <ref name="message-id-attribute"/>
   <element name="data">
    <interleave>
     <grammar ns="http://example.com/ns/dhcp">
      <include href="dhcp-gdefs.rng"/>
      <start>
       <optional>
        <element name="dhcp:dhcp">
         <interleave>
          <optional>
           <element name="dhcp:max-lease-time">
            <data type="unsignedInt"/>
           </element>
          </optional>
          <optional>
           <element name="dhcp:default-lease-time">
            <data type="unsignedInt"/>
           </element>
          </optional>
          <ref name="_dhcp__subnet-list"/>
          <optional>
           <element name="dhcp:shared-networks">
            <zeroOrMore>
             <element name="dhcp:shared-network">
              <element name="dhcp:name">
               <data type="string"/>
              </element>
              <ref name="_dhcp__subnet-list"/>
             </element>
            </zeroOrMore>
           </element>
          </optional>
          <optional>
           <element name="dhcp:status">
            <zeroOrMore>
             <element name="dhcp:leases">
              <element name="dhcp:address">
               <ref name="ietf-inet-types__ip-address"/>
              </element>
              <interleave>
               <optional>
                <element name="dhcp:starts">
                 <ref name="ietf-yang-types__date-and-time"/>
                </element>
               </optional>
               <optional>
                <element name="dhcp:ends">
                 <ref name="ietf-yang-types__date-and-time"/>
                </element>
               </optional>
               <optional>
                <element name="dhcp:hardware">
                 <interleave>
                  <optional>
                   <element name="dhcp:type">
                    <choice>
                     <value>ethernet</value>
                     <value>token-ring</value>
                     <value>fddi</value>
                    </choice>
                   </element>
                  </optional>
                  <optional>
                   <element name="dhcp:address">
                    <ref name="ietf-yang-types__phys-address"/>
                   </element>
                  </optional>
                 </interleave>
                </element>
               </optional>
              </interleave>
             </element>
            </zeroOrMore>
           </element>
          </optional>
         </interleave>
        </element>
```

```
        </optional>
       </start>
      </grammar>
     </interleave>
    </element>
   </element>
  </start>
 </grammar>
```

### C.3.2. Схема RELAX NG - определения глобально именуемых шаблонов

```
<?xml version="1.0" encoding="utf-8"?>
<grammar
    xmlns="http://relaxng.org/ns/structure/1.0"
    xmlns:nma="urn:ietf:params:xml:ns:netmod:dsdl-annotations:1"
    xmlns:dhcp="http://example.com/ns/dhcp"
    datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
 <define name="ietf-yang-types__phys-address">
  <data type="string">
   <param name="pattern">
    ([0-9a-fA-F]{2}(:[0-9a-fA-F]{2})*)?
   </param>
  </data>
 </define>
 <define name="ietf-inet-types__ipv6-address">
  <data type="string">
   <param name="pattern">... шаблон regex ...</param>
  </data>
 </define>
 <define name="ietf-inet-types__ip-prefix">
  <choice>
   <ref name="ietf-inet-types__ipv4-prefix"/>
   <ref name="ietf-inet-types__ipv6-prefix"/>
  </choice>
 </define>
 <define name="ietf-inet-types__host">
  <choice>
   <ref name="ietf-inet-types__ip-address"/>
   <ref name="ietf-inet-types__domain-name"/>
  </choice>
 </define>
 <define name="ietf-yang-types__date-and-time">
  <data type="string">
   <param name="pattern">... шаблон regex ...</param>
  </data>
 </define>
 <define name="_dhcp__subnet-list">
  <zeroOrMore>
   <element name="subnet">
    <element name="net">
     <ref name="ietf-inet-types__ip-prefix"/>
    </element>
    <interleave>
     <optional>
      <element name="range">
       <interleave>
        <optional>
         <element name="dynamic-bootp">
          <empty/>
         </element>
        </optional>
        <element name="low">
         <ref name="ietf-inet-types__ip-address"/>
        </element>
        <element name="high">
         <ref name="ietf-inet-types__ip-address"/>
        </element>
       </interleave>
      </element>
     </optional>
     <optional>
      <element name="dhcp-options">
       <interleave>
        <zeroOrMore>
         <element name="router">
          <ref name="ietf-inet-types__host"/>
         </element>
        </zeroOrMore>
        <optional>
         <element name="domain-name">
          <ref name="ietf-inet-types__domain-name"/>
         </element>
        </optional>
       </interleave>
      </element>
     </optional>
     <optional>
```

```
        <element name="max-lease-time">
         <data type="unsignedInt"/>
        </element>
       </optional>
      </interleave>
     </element>
    </zeroOrMore>
   </define>
   <define name="ietf-inet-types__domain-name">
    <data type="string">
     <param name="pattern">... шаблон regex ...</param>
     <param name="minLength">1</param>
     <param name="maxLength">253</param>
    </data>
   </define>
   <define name="ietf-inet-types__ipv4-prefix">
    <data type="string">
     <param name="pattern">... шаблон regex ...</param>
    </data>
   </define>
   <define name="ietf-inet-types__ipv4-address">
    <data type="string">
     <param name="pattern">... шаблон regex ...</param>
    </data>
   </define>
   <define name="ietf-inet-types__ipv6-prefix">
    <data type="string">
     <param name="pattern">... шаблон regex ...</param>
     <param name="pattern">... шаблон regex ...</param>
    </data>
   </define>
   <define name="ietf-inet-types__ip-address">
    <choice>
     <ref name="ietf-inet-types__ipv4-address"/>
     <ref name="ietf-inet-types__ipv6-address"/>
    </choice>
   </define>
  </grammar>
```

### C.3.3. Схема Schematron для отклика <nc:get>

```
  <?xml version="1.0" encoding="utf-8"?>
  <sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron">
   <sch:ns uri="http://example.com/ns/dhcp" prefix="dhcp"/>
   <sch:ns uri="urn:ietf:params:xml:ns:netconf:base:1.0" prefix="nc"/>
   <sch:pattern abstract="true" id="_dhcp__subnet-list">
    <sch:rule context="$start/$pref:subnet">
     <sch:report test="preceding-sibling::$pref:subnet
                       [$pref:net=current()/$pref:net]">
      Дубликат ключа net
     </sch:report>
    </sch:rule>
    <sch:rule
      context="$start/$pref:subnet/$pref:dhcp-options/$pref:router">
     <sch:report test=".=preceding-sibling::router">
      Дубликат значения leaf-list "<sch:value-of select="."/>"
     </sch:report>
    </sch:rule>
   </sch:pattern>
   <sch:pattern id="dhcp">
    <sch:rule
      context="/nc:rpc-reply/nc:data/dhcp:dhcp/dhcp:default-lease-time">
     <sch:assert test=". &lt;= ../dhcp:max-lease-time">
      default-lease-time должно быть не больше max-lease-time
     </sch:assert>
    </sch:rule>
    <sch:rule context="/nc:rpc-reply/nc:data/dhcp:dhcp/
                       dhcp:shared-networks/dhcp:shared-network">
     <sch:report test="preceding-sibling::dhcp:shared-network
                       [dhcp:name=current()/dhcp:name]">
      Дубликат ключа dhcp:name
     </sch:report>
    </sch:rule>
    <sch:rule context="/nc:rpc-reply/nc:data/dhcp:dhcp/
                       dhcp:status/dhcp:leases">
     <sch:report test="preceding-sibling::dhcp:leases
                       [dhcp:address=current()/dhcp:address]">
      Дубликат ключа dhcp:address
     </sch:report>
    </sch:rule>
   </sch:pattern>
   <sch:pattern id="id2768196" is-a="_dhcp__subnet-list">
    <sch:param name="start" value="/nc:rpc-reply/nc:data/dhcp:dhcp"/>
    <sch:param name="pref" value="dhcp"/>
   </sch:pattern>
   <sch:pattern id="id2768215" is-a="_dhcp__subnet-list">
    <sch:param name="start"
```

```
              value="/nc:rpc-reply/nc:data/dhcp:dhcp/
                     dhcp:shared-networks/dhcp:shared-network"/>
    <sch:param name="pref" value="dhcp"/>
   </sch:pattern>
  </sch:schema>
```

### *C.3.4. Схема DSRL для отклика <nc:get>*

```
  <?xml version="1.0" encoding="utf-8"?>
  <dsrl:maps
      xmlns:dsrl="http://purl.oclc.org/dsdl/dsrl"
      xmlns:dhcp="http://example.com/ns/dhcp"
      xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
   <dsrl:element-map>
    <dsrl:parent>/nc:rpc-reply/nc:data</dsrl:parent>
    <dsrl:name>dhcp:dhcp</dsrl:name>
    <dsrl:default-content>
     <dhcp:max-lease-time>7200</dhcp:max-lease-time>
     <dhcp:default-lease-time>600</dhcp:default-lease-time>
    </dsrl:default-content>
   </dsrl:element-map>
   <dsrl:element-map>
    <dsrl:parent>/nc:rpc-reply/nc:data/dhcp:dhcp</dsrl:parent>
    <dsrl:name>dhcp:max-lease-time</dsrl:name>
    <dsrl:default-content>7200</dsrl:default-content>
   </dsrl:element-map>
   <dsrl:element-map>
    <dsrl:parent>/nc:rpc-reply/nc:data/dhcp:dhcp</dsrl:parent>
    <dsrl:name>dhcp:default-lease-time</dsrl:name>
    <dsrl:default-content>600</dsrl:default-content>
   </dsrl:element-map>
   <dsrl:element-map>
    <dsrl:parent>
     /nc:rpc-reply/nc:data/dhcp:dhcp/dhcp:subnet
    </dsrl:parent>
    <dsrl:name>dhcp:max-lease-time</dsrl:name>
    <dsrl:default-content>7200</dsrl:default-content>
   </dsrl:element-map>
   <dsrl:element-map>
    <dsrl:parent>
     /nc:rpc-reply/nc:data/dhcp:dhcp/dhcp:shared-networks/
     dhcp:shared-network/dhcp:subnet
    </dsrl:parent>
    <dsrl:name>dhcp:max-lease-time</dsrl:name>
    <dsrl:default-content>7200</dsrl:default-content>
   </dsrl:element-map>
  </dsrl:maps>
```

#### Адрес автора

**Ladislav Lhotka** (editor)
CESNET
EMail: lhotka@cesnet.cz

#### Перевод на русский язык

Николай Малых

nmalykh@protokols.ru